AD-A056 055 OFFICE OF THE COMPTRO_LEF (NAVY) WASHINGTON DC F/G 9/2
MINIGAP GENERALIZED ANALYSIS PACKAGE. A TOOL FOR AIDING MANAGEM--ETC(U)
APR 78 S MASIELLO

UNCLASSIFIED MGAP-PG-01 DOD/DF-78/003A NL

1 OF 2
AD
A056055

MINIGAP

Generalized Analysis Package

A Tool for aiding management in analysis of large data bases

PROGRAMMER'S MANUAL

DEPARTMENT OF THE NAVY
OFFICE OF THE COMPTROLLER

PROGRAMMER'S DOCUMENTATION GUIDE

TABLE OF CONTENTS

BIBLIOGRAPHIC DATA SHEET

1. Report No. MGAP-PG-01 18 DOD/DF-19 78/003a

2.

3. Recipient's Accession No.

4. Title and Subtitle

6 MINIGAP Generalized Analysis Package, PROGRAMMER'S ~~GUIDE~~ Manual, A Tool for Aiding Management in Analysis of Large Data Bases,

5. Report Date
4-20-78

6.
Date of Approval

7. Author(s)
10 Sheryl Masiello

8. Performing Organization Rept. No.

9. Performing Organization Name and Address

Office of the Comptroller of the Navy
Office of Financial Analysis (NCD-5)
Crystal Mall 2 Room 602
Washington, DC 20350

10. Project/Task/Work Unit No.
NA

11. Contract/Grant No.
NA

12. Sponsoring Organization Name and Address
NA

12 111 p.

13. Type of Report & Period Covered
NA

14.

15. Supplementary Notes

For magnetic tape, see

16. Abstracts

This document is intended to help programmers understand the software of the MINIGAP report generation and generalized analysis package. It includes flow charts, descriptions of the input files, descriptions of the subroutine modules, and cross-indexing of the subroutine calls. This guide should enable a FORTRAN programmer to understand the MINIGAP system well enough to modify the system as desired. The MINIGAP USER'S MANUAL is necessary to understand how to use the system. The USER'S MANUAL and the PROGRAMMER'S MANUAL should be supplied with each MINIGAP system.

11 20 Apr 78

17. Key Words and Document Analysis. 17a. Descriptors

Thesaurus of Engineering and Scientific Terms and 1964 COSATI Subject Category List not available. See Abstracts.

14 MGAP-PG-01

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement
Release Unlimited

19. Security Class (This Report)
UNCLASSIFIED

20. Security Class (This Page)
UNCLASSIFIED

21. No. of Pages
110

22. Price

A056 056

GENERAL

MINIGAP is a modular FORTRAN system which allows users to design their own report formats fairly flexibly. It is a general system, in that it allows users to access different types of data, in different databases, and to formulate new reports at run-time.

It is a set of 34 FORTRAN modules. There is a main routine, a block data subroutine, five report generating subroutines and 26 supporting subroutines. REPØ1Ø generates a report which is specific to NCD-5. Since this report is very specific, it would be of little use to someone else. No user documentation is provided for this report, and there is little programming documentation provided.

A pictorial description of the system is provided in the section 'FLOW OF CONTROL.'

This system is installed on the INTERDATA 7/32. The computer system configuration contained 256 K bytes of core storage at installation. Using overlays, squeeze options on compilations, and byte-programming (as necessary), we were able to set up this system with 167 K bytes. This does not allow room for any other systems to operate at the same time on the INTERDATA.

Three input files are necessary to run MINIGAP. One of these is the input parameter file, set up by the user. This is described in the user's documentation guide. A driver file, which contains organization and account codes, is also necessary. This file is described in one of the following sections 'DRIVER FILE'. The other input file contains the data for the organization. This is described in the next section, 'MASTER FILE.'

Only one output file is necessary. Device #15 should be assigned to this file. All error messages and system generated responses will be sent to this file. If the parameter 'OUTPUT' is equated to another device (such as #14), the report will be sent to this device, instead of device #15. This will separate the report from the MINIGAP system messages.

A limited number of devices are available to the user. The description of the overlays is in the section titled 'OVERLAYS.'

SECTION 2. INPUT FILES

MASTER FILE

The master file contains all the data used by MINIGAP. It contains account data for each organization, for each time period. It is the most difficult file to work with, since some special facility (such as a database management system) must be available to create and update it. MINIGAP only reads from the file to get the data for the reports. It does not allow a user to access the master file directly, nor write to it.

The entire file is a fixed length, binary file. It is accessed using random access I/O. It must contain m records, where:

m = (number of organizations x number of time
    periods) + 1.

Each record will contain double precision cumulative data for each account and the account codes. There is one record per organization per time period.

The first record in the file does not contain data. It contains information for the time periods. This first record must be long enough to contain all the information for all the time periods. Since the file is a fixed length file, the record length must be long enough to accommodate the time periods in the first record, and the accounts in the remaining records. To ensure these conditions, the following equations must hold:

number of accounts = ((record length/4)-4)/3
number of time periods = ((record length/8)-24)/5

The master file should be set up with a record for each organization. Organizations include: all organizations at level 1, the subtotals at levels 2 and 3, and the overall total at level 4. Therefore, some sophisticated method of data entry is required. This method must be able to create the subtotals, and to update the necessary subtotals when a change is made in a level 1 organization. Although this creates more work at data entry, it allows a faster access time since each subtotal is saved and doesn't need to be recalculated for every request.

The master file is arranged in blocks of data. Each block of records represents one time period. Each of the records within a block represents data for one organization. One record will contain all the data for all the accounts for one organization one time period. The layout of an individual record containing data for n accounts is:

Organization Name	Time Period	UIC Code	Account Codes	Account Data
8 bytes	2 bytes	6 bytes	n x 4 bytes	n x 8 bytes

All the records concerning one time period will be stored together. Within any time period block, the organizational data should be in the same order as within the other time period blocks. The driver file will contain a number representing the organization's relative position in a block. The beginning of a time period block is stored in the first record of the master file. Adding these two numbers will give the location of the data for an organization at a given time period. The next 6 paragraphs explain the first record of the master file. These paragraphs describe the content and form of the record, and the purpose of the information.

The first record of the file contains the information for the time periods. The first 2n words (where n = the number of time periods) will contain the 8 character names of each time period.

The next ½ n words will contain a number pointing to the first record of each time period's block of data. These numbers, each contained in ½ word, should be in the same order as the time period names. Since the first record is not used for data, the record number at the start of the first time period will equal 2.

The next ½ n words will point to the time periods, if any, which would contain data for the month previous to this time period. When EXTRCT is used to extract the data from the master file, it will use this pointer (if required) to retrieve data from the previous month time period. It will subtract this data from the other data retrieved to obtain a monthly amount. (Note: the pointer does not have to point to the time period for the previous month. However, EXTRCT will still perform the above subtraction if the data type is set to "M" in the input file.)

The next ½ n words are the same as above, except that they are used to point to time periods for the previous quarter. (The same NOTE applies here, also, for data type "Q".)

The next ½ n words will point to another time period, as determined by the user/programmer. These will enable a user to enter a specific time period; if the user specifies data type "P", the data for the time period pointed to by this pointer is used. One possible use would be to have this pointer point to the time period containing data from the previous year at this time.

The next ½ n words are used in the same manner as the previous ½ n words. A possible use for these pointers could be to point to the time period from the base year at this time. This pointer would be used if the data type was given as "L".

Any or all of these pointers could be ignored, and set to zero. This would only allow cumulative data to be accessed directly by the user, however.

MINIGAP is set up to accommodate up to 350 accounts , 160 organizations and 100 time periods. These bounds may be changed in the dimensions of the system. The size of the master file is determined by the following equation:

$$\text{SIZE (in bytes)} = (\text{\# of organizations X \# of time periods}) \text{ X record length.}$$

However, the size of a file which reaches each of these maximums would be 67 megabytes. The disk currently used with this Interdata can store up to 25 megabytes; however, only about 20 megabytes are free for user storage. Therefore, no data file of this size has been used with MINIGAP, although files which reach one or two of the maximums have been used successfully. The size of the data base should be less than 20 megabytes, and none of the bounds (as mentioned previously) should be exceeded. (Note: the dimensions will really only allow 348 accounts. If 350 accounts are desired, all occurrences of '348' must be changed to '350' in the MINIGAP system.)

These bounds were chosen to accommodate working databases. An enlargement of any of these will necessarily enlarge the amount of core needed to run MINIGAP. Since MINIGAP requires most of the available core of the Interdata system (which has 256k core), these boundaries were not raised. They may be raised within physical limits, requiring only dimension changes. However, if more than 400 organizations are needed (including all subtotals from the four levels), logic changes to NXTKEY and the user documentation would be required.

DRIVER FILE

This file determines how the master file (data file) is accessed. More than one driver file could exist for the same master file. It contains information about the organization hierarchy, the organization codes, and the account codes.

One of the main restrictions concerning the data structure is that it must be hierarchical. Within this, only four levels of hierarchy are allowed. The organizations are accessed from the bottom to the top. The organizations at level-1 cannnot be broken down into any sub-groupings. These organizations are arranged, according to the user's instructions, into groups. Level 2 contains subtotals. All the organizations within group 1 are totalled into the level-2 subtotal for group 1, and the same for any of ther other groups. These level-2 subtotals can then be consolidated into level-3 subtotals, in the same manner. Level-4 will contain the total of all the organizations. Each of these organizations and subtotals are numbered in the driver file. This provides the order of output. For example, the first organization output for any group will always be the one indexed with the number 1.

Because of the size restructions on MINIGAP, the number of subdivisions had to be restricted. This otherwise arbitrary restriction limits the number organizations within each group to 19, with the subtotal being the 20th organization. This structure is also built into the function NXTKEY. Changing this restriction would mainly involve changes to NXTKEY, the drive file, and the user documentation.

The first line of the driver file is used to specify the record length used in the master file. This number is used to calculate the number of accounts possible and the number of time periods possible. This number is right justified in the first five spaces.

The following n lines contain the organization data. There is a line for each organization, and each subtotal. The following information is contained on each line:

1. A unique 8-character alphanumeric name for the organization or subtotal, followed by one blank.

2. A two digit number not to exceed 20, specifying the organization's rank at level-1.

3. A two digit number, not to exceed 20, which specifies the subtotal, or level-2 group, of which the organization is a member. (The combination of this number with the previous number must uniquely specify the organization or subtotal.)

4. A two digit number, not to exceed 20, which specifies the subtotal, or level-3 group of which the organization is a member. This is followed by two blanks.

5. A six digit alphanumeric, unique identifier code for the organization or subtotal, (This is not presently used, but the information is there if necessary.)

6. A three digit number which specifies the relative location of the data for this organization or subtotal within the master file. This number specifies what data will be accessed when this organization is requested. This followed by one blank.

7. The next 44 spaces are used for the complete name of the organization, and may include any valid symbols.

Since all access of the master file is based on this file, it can easily be seen how an organization may be deleted or moved within the hierarchy, without changing the master file. Removal of the organization from the driver file essentially removes it from the user's access. Changing items 2, 3 or 4 (above) can be used to "move" the organization within the hierarchy.

After the lines for the organizations, the following line is included as a delimiter: '0999'. The lines following this line refer to the accounts. There is a line for each account. The account codes listed here must include all the valid accounts. If there is data for an account in the master file, and the account code is not listed in the driver file, that data cannot be accessed using MINIGAP. The following information is contained on each of the account lines:

1. The complete name of the account, using 36 characters.

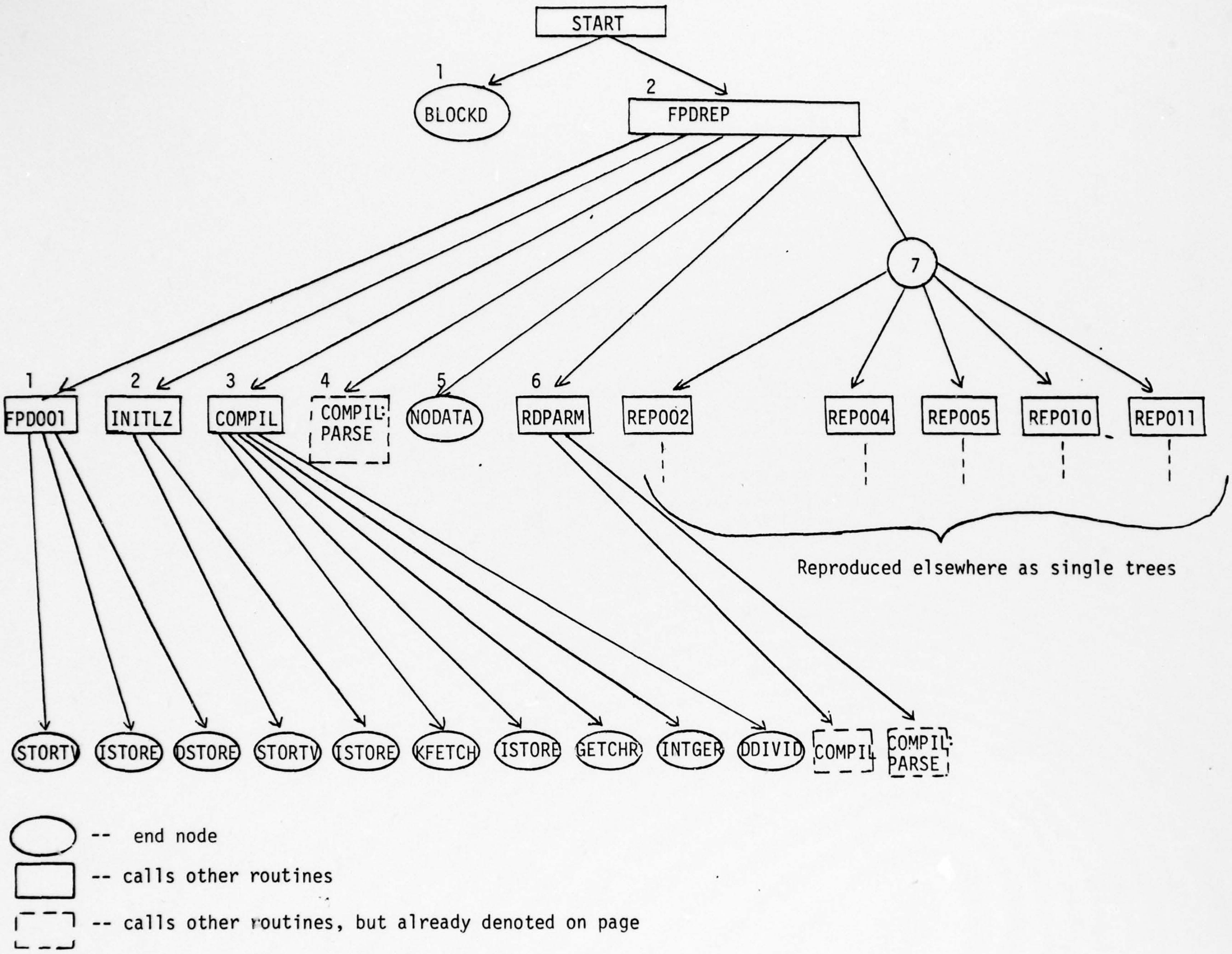
2. A four digit alphanumeric account code. (The routine 'SPREAD', used to spread data over several periods in REP005, REP010, and REP011, uses methods which are not reasonable for all types of data. This routine keys on the first two digits of the account code. Therefore, development of the account codes and/or modifications of SPREAD should be coordinated. For further information, see the specification sheets for the routine SPREAD.)

The last line in the file should be the line: '0999'. The driver file must be set up manually. MINIGAP expects it to contain the above information, and uses it as such, without providing any checking.

Since this sets up the organization hierarchy, account coding, etc., the development of the driver file/s should be coordinated with the user/s.

SECTION 3. DESCRIPTION OF REPORT MODULES

MINIGAP


START
1
BLOCKD
2
FPDREP
7
1
FPD001
2
INITLZ
3
COMPIL
4
COMPIL:
PARSE
5
NODATA
6
RDPARM
REP002
REP004
REP005
REP010
REP011
Reproduced elsewhere as single trees
STORTV
ISTORE
DSTORE
STORTV
ISTORE
KFETCH
ISTORE
GETCHR
INTGER
DDIVID
COMPIL
COMPIL:
PARSE
-- end node
-- calls other routines
-- calls other routines, but already denoted on page

MAIN

1. The main routine, FPDREP, calls FPDØØ1. This subroutine is used to read the driver file. It sets up all the tables and indices using the STORTV routines, to allow easy access to the information contained in the driver file. The information is stored so that all the information for one organization can be accessed using the same index.

2. IFETCH, a system utility, is used to fetch the overlay containing INITLZ and RDPARM (see the section OVERLAYS). INITLZ is now called to store the items initialized in the block data routine and the account names read by FPDØØ1 in the array T1.

3. The logical units containing the overlays are now rewound, so they may be accessed more than once. COMPIL is now called to read the first line of the input file. This line is assumed to give the report type. COMPIL only checks for a number following an '!'. This number is assumed to specify the report type, and is passed back via the variable LINE. (LINE does not necessarily equal the number.) If an end-of-file is reached, signifying that there are no more reports requested, FPDREP generates an 'END OF REPORT PROCESSING' message and stops.

4. PARSE is immediately called to translate the results of COMPIL. The number is now stored in the variable VALUE. FPDREP outputs a message to device #15, specifying the report type requested.

5. NODATA is called to initialize its variables for future use.

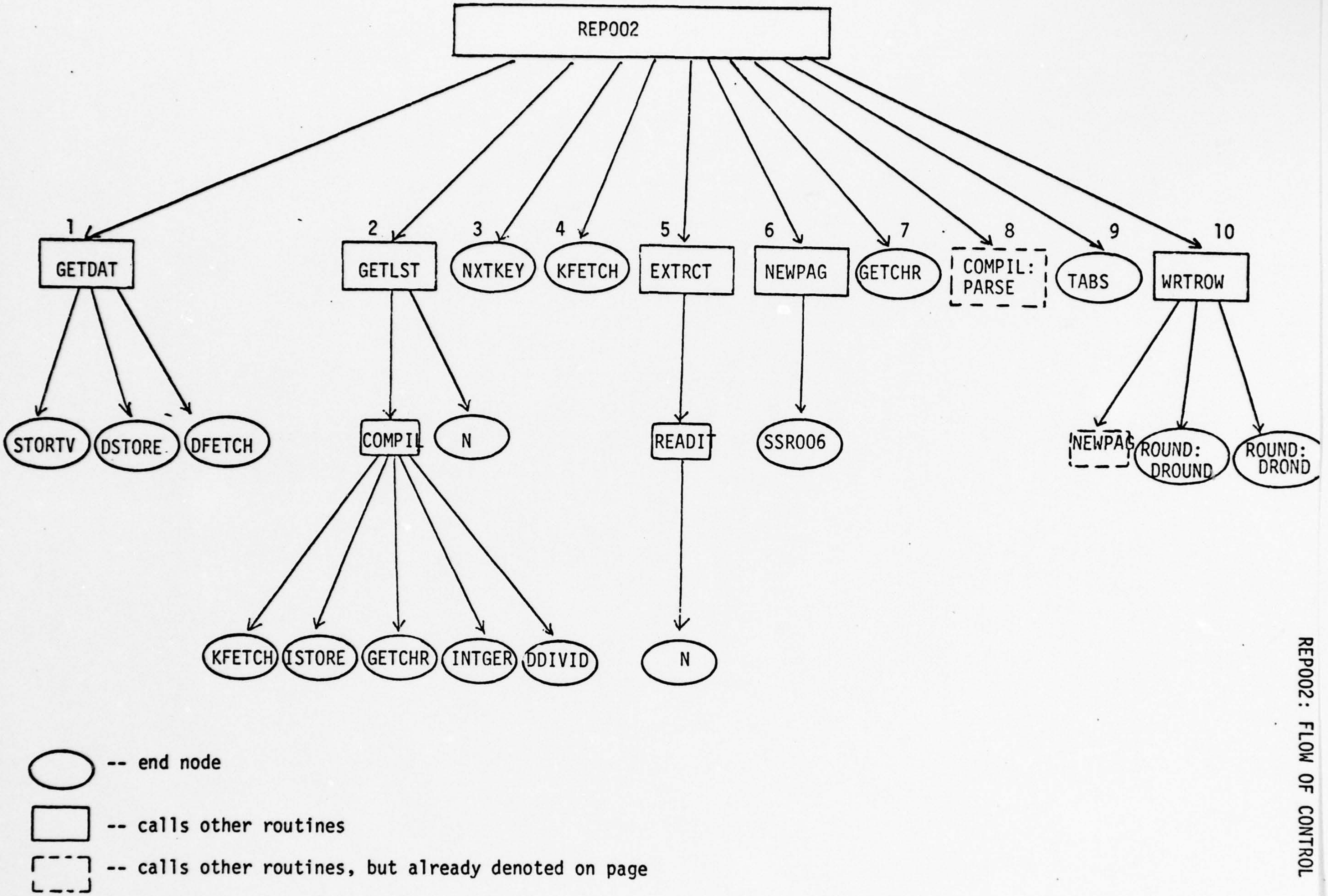
6. IFETCH is called again, to fetch the overlay which contains RDPARM.

6a. RDPARM is now called. It will read the input parameters line by line, until is reaches a line starting with 'xx'. It will read one line after this, which it assumes to be the title of the report. If an end-of-file is reach in any of its reads, an error message is output to device #15 and the MINIGAP run stops. It will compare the value given for each parameter with its range (as specified in the blockdata subroutine). If it exceeds its range, RDPARM will output an error message to device #15 and set the parameter to its default value. (Any parameters which are not specified have already been set to their default value in INITLZ.)

7. FPDREP now uses a computed GOTO to fetch the desired report type. If the report type is not valide, an error message is output to devide #15 and the MINIGAP run stops. Otherwise, the desired report generator is called. When control returns to FPDREP, it checks to see if any more reports are desired. (Control goes to step #3)

M I N I G A P

REP002: FLOW OF CONTROL



REP002

1. When control is given to this report generator, GETDAT is called. Since GETDAT reads the time period required for the report, the next line in the input file must specify the time period. If this line does not contain a valid time period, an error message is sent to device #15 and the MINIGAP run stops. If the time period is valid, GETDAT sets up the variables which will later be used by EXTRCT to reference the correct data in the master file. At this time, GETDAT outputs to device #15 its translation of the data type requested (in month, year and data type).

2. In this report, only one time period may be specified. After this one line has been read (step #1, above), GETLST is called to read the remaining lines of the input file. These lines should specify the rows of the report. GETLST will continue reading lines until it reaches a line beginning '0999', or the end-of-file, or more than 50 accounts and calculations have been read. In the latter case, an error message is output to device #15, and the MINIGAP run stops. Otherwise, control is passed back to the report generator, with indices to the accounts and computations stored in the array EXP. These will eventually be used to output the rows of the report. The array INDEX will also contain a code to specify the row. If an account is requested, INDEX will be assigned a number relative to the account. (Since there are N accounts, these numbers will range from 1 to N.) If a calculation is desired the number assigned will be greater than N, but less than 900. Format requests, such as paging, underlining, etc., will receive codes between 900 and 910.

3. The report generator row calls the function NXTKEY to get the relative location of the next requested organization. It will then check to see if the new "activity" is in the same "command" as the previous (if any) "activity". If it is not, control is sent to NEWPAG (step #5).

3a. Otherwise, the variable KEY which was assigned by NXTKEY, is checked. If it equals zero (no more organizations), this report is finished and control returns to FPDREP (main).

4. EXTRCT is now called to load the data for the organization into the array DATA from the master file. The organization short-name, which is in the master file, is loaded into the array TITLE, to serve as the column heading. The report generator now requests the next organization (step #3).

5. In this section, NEWPAG is called to write the report header and the column headings. The variable INDEX is now checked to determine the contents of the row. If the row is a formatting request (INDEX-900), control is given to WRTROW (step #8). If the row contains strictly account data (no calculations), the next section (step #6) is skipped.

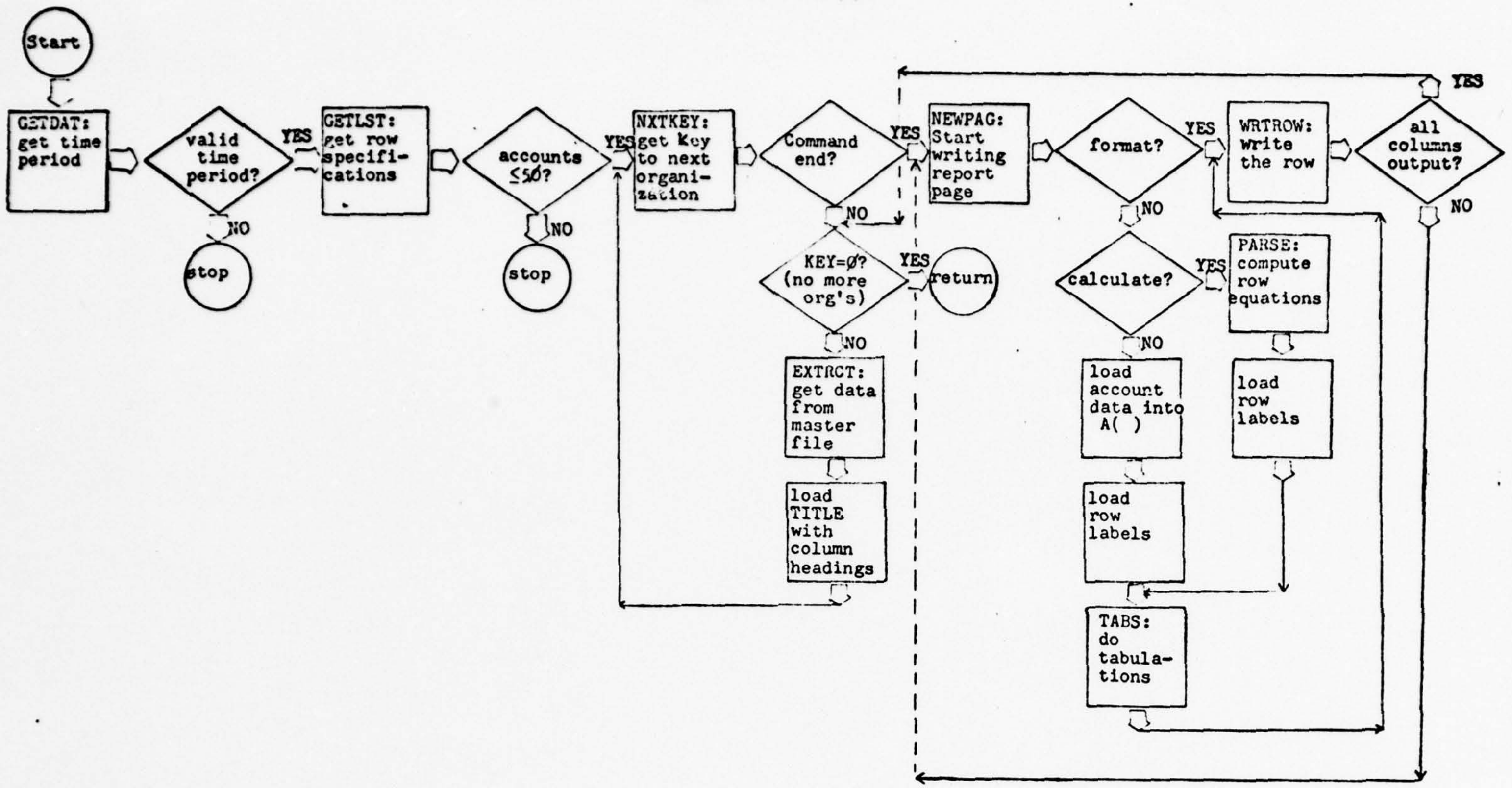
6. PARSE is now called to compute any row calculations, using the array EXP, which was previously set up by GETLST. The user-specified label for the computation is now loaded into the variable INAME. The calculated data is loaded into the array A. Control is now given to TABS (step #7a).

7. The account data is now stored in the variable A, according to the account requested. The account code is now stored in INAME, for the row label.

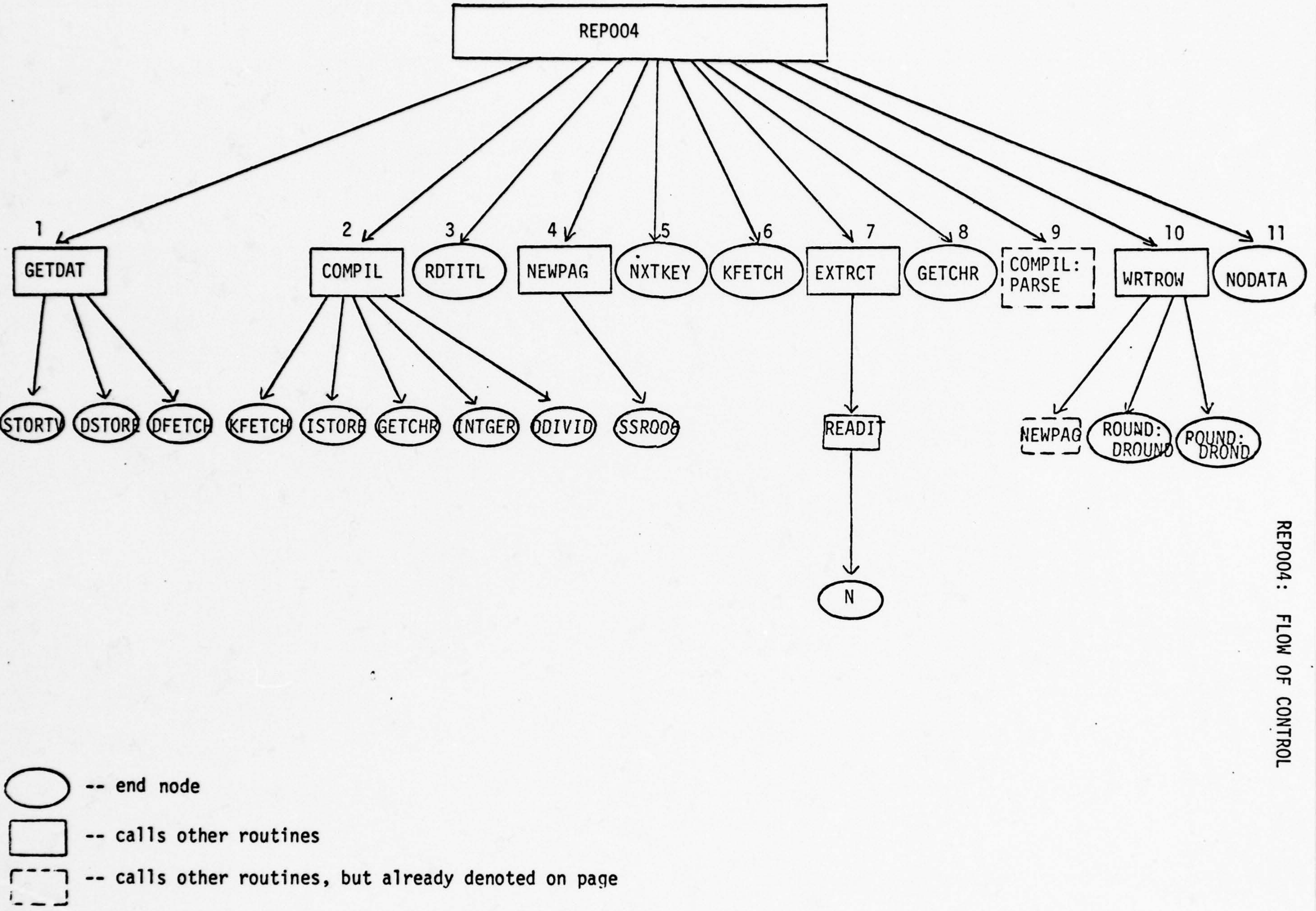
7a. TABS is now called. It will skip blank lines on the report to perform the vertical tabbing. It will load the row labels (INAME) into JNAME, and will insert blanks into JNAME to perform the horizontal tabbing. This tabbing will be executed in blocks of four. (For a tab of 2, eight blanks will be inserted.)

8. WRTROW is called to output the row onto the report. Any format specifications requested in the input parameter list will be used by WRTROW. It will load commas as necessary, output the correct number of decimal places, etc. After WRTROW has returned control (the row has been written), the variable L is checked to determine whether all the columns for that "command" have been written. If not, control is returned to NEWPAG (step #5). If all the columns have been written, the report generator checks to see if any more organizations have been specified. This is done by checking the variable KEY. (Control is returned to step #3a.)

FLOW OF REPØØ2



REP004: FLOW OF CONTROL



REP004

1. When control is given to this report generator, TSIZE is set equal to the number of accounts involved for this database. GETDAT is then called to read the time period from the input list. If this line does not contain a valid time period, an error message is output to device #15 and the MINIGAP run stops. If the time period is valid, GETDAT sets up the variables which will later be used by EXTRCT to reference the correct data. At this time, GETDAT outputs its translation of the time period requested (in month, year and data type) to device #15.

2. In this report, only one time period may be specified. After this one line has been read (setp #1, above) COMPIL is called. COMPIL reads a line which is assumed to specify the column contents. This line should equate a column with either an account, or an account calculation. The format used is: Cn = xxxx where n = the appropriate column number, and xxxx represents an account code, or an account calculation. The report generator will continue calling COMPIL, once for each column, until all the columns are specified or there is no more input. (If the latter is the case, an error message should be generated; however, this does not happen.)

3. RDTITL is now called to read the next n (number of columns) lines. These should contain the column headings, as they will be treated as column headings. These will be loaded into the appropriate positions of the array TITLE, and will be centered. If an end-of-file is reached before all the columns have been titled, an error message results. These columns will not have headings, though.

4. All the input has now been processed, and NEWPAG is now called to write the heading for the report.

5. the function NXTKEY is now used to set KEY to the relative location of the next organization requested. If this equals zero, there are no more organizations required, and control passes to the last section (step #10). At the beginning of a new "command" group (IROW=1), ERRFLAG and FLAG are initialized to .false. (However, to avoid any possibility of error, these should also be initialized at the start of this report.)

6. KFETCH is now called to obtain the index which points to the location of KEY in K1. (This will be used by NODATA, if it is necessary to call NODATA for this organization.) If the KEY is nout found, the next KEY is requested (returning to step #5).

7. EXTRCT is called to read the correct data from the master file into the variable DATA. If no data is present for this activity, control is passed to NODATA (step #7a). Otherwise, control passes to the next section (step #8).

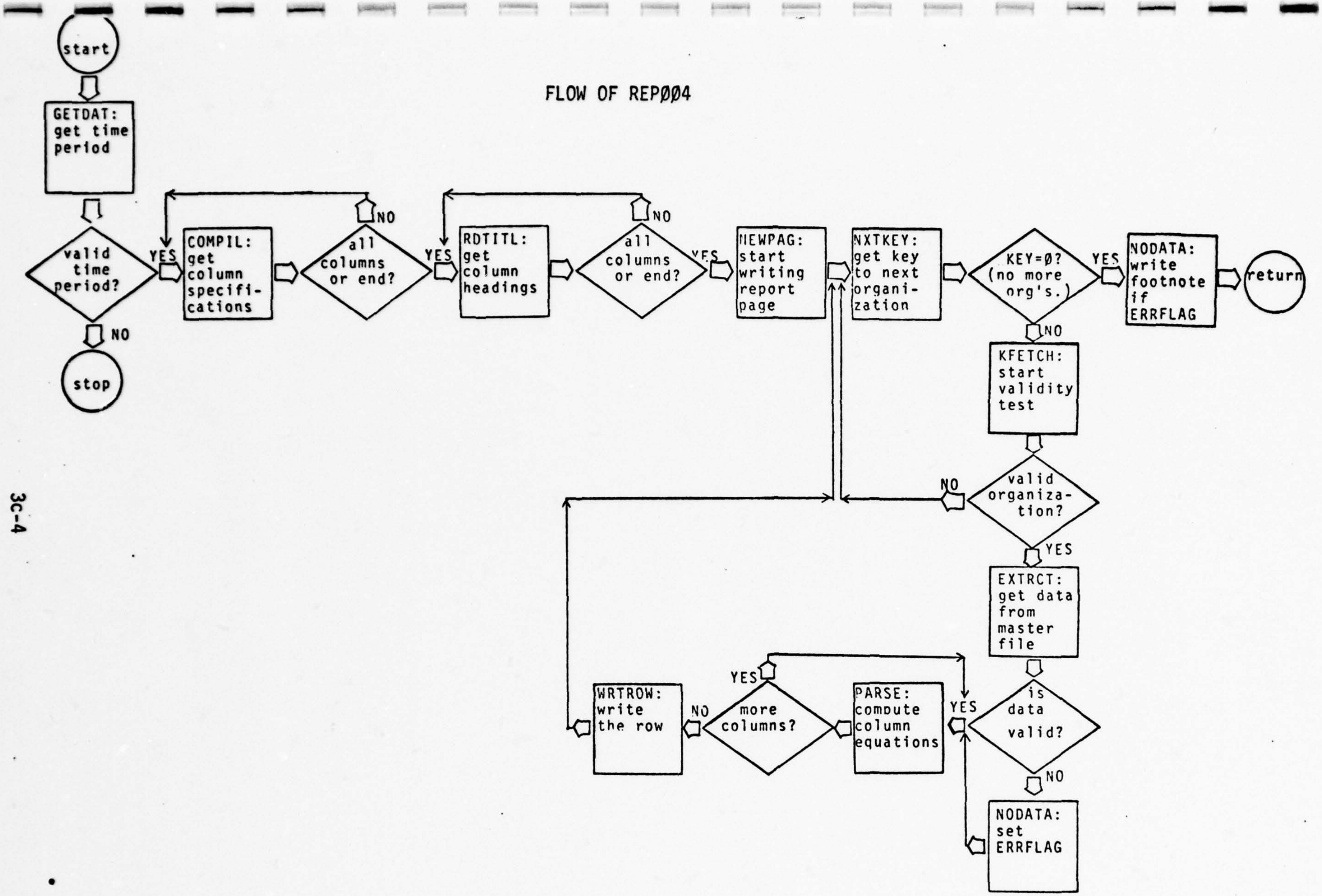
7a. NODATA is called using the index obtained from KFETCH. ERRFLAG is also set to .true. to show that at least one activity in that "command" has no data.

8. All the data returned by EXTRCT is now loaded into areas of the array SPACE, based on the account codes. PARSE is now called to perform any column calculations, and put the accounts in the right columns. The data is now stored in the array A. PARSE is called for each column.

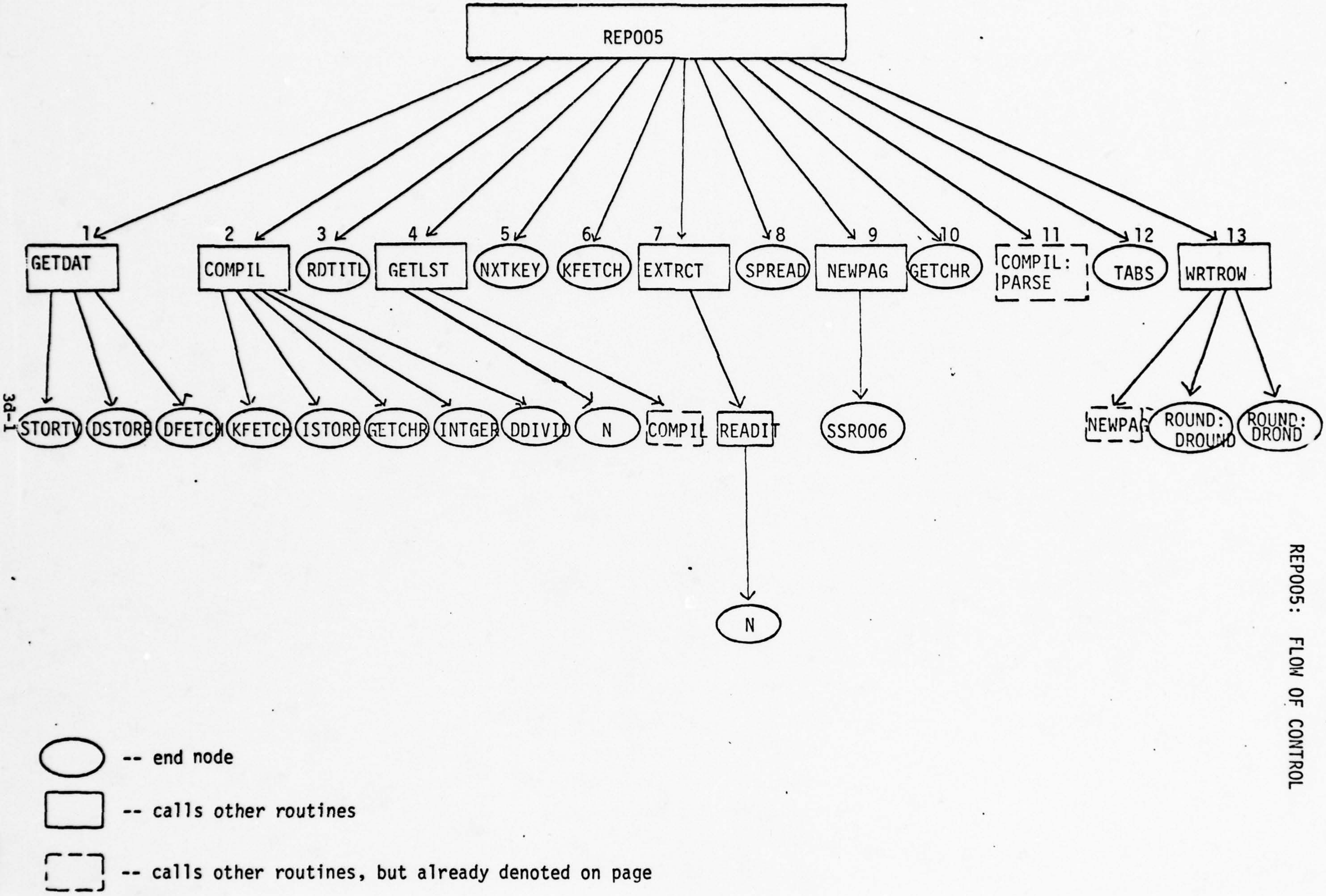
9. If the organization represents a sub-total, IFLAG is set to mark the row off with hased lines. In addition, if any data was missing from this subtotal (ERFLAG - .true.), FLAG is set to output an asterisk to the left of the row. WRTROW is now called to write this row. The next organization is then fetched, using NXTKEY (control returns to step #5).

10. NODATA is now called. This step is reached when the report is finished. NODATA will write a message (footnote) to the report if data for any of the reported organizations (or sub-totals) was missing. It will also reset its variables for the next report. Since the report is now finished, control is returned to FPDREP (main).

FLOW OF REPØØ4


start
GETDAT: get time period
valid time period?
YES
NO
stop
COMPIL: get column specifi- cations
all columns or end?
NO
YES
RDTITL: get column headings
all columns or end?
NO
YES
NEWPAG: start writing report page
NXTKEY: get key to next organi- zation
KEY=Ø? (no more org's.)
YES
NODATA: write footnote if ERRFLAG
return
NO
KFETCH: start validity test
valid organiza- tion?
NO
YES
EXTRCT: get data from master file
is data valid?
YES
NO
NODATA: set ERRFLAG
PARSE: compute column equations
more columns?
YES
NO
WRTROW: write the row

MINIGAP



REP005: FLOW OF CONTROL

REP005

1. When control is given to this report generator, TSIZE is set equal to the number of accounts in the database. GETDAT is then called to read the time period/s for this report. The maximum number of time periods used in this report is equal to the number of columns. GETDAT will read the next n lines of the input file as time periods, (when n = the number of columns) unless it reaches a line beginning with '0999' (which signifies the end of the time period specifications), or an invalid time period, or an end-of-file. Each of these cases terminates the reading done by GETDAT. The last two cases will cause an error message to be sent to device #15, and the MINIGAP run will stop. For each correct time period input, GETDAT sets up the variables which will later be used by EXTRCT to reference the correct data. GETDAT will also output its translation of each time period requested (in month, year and data type).

2. COMPIL is now called, to read the column specifications. It will read one line for each column, unless it reaches the end-of-file. If there are too few column specifications, the output for the remaining columns will be underfined. However, no error message is output to bring the user's attention to this input problem, and the run continues. (An error message will be output by RDTITL, though when it too reaches an unexpected end-of-file.) Since column calculations are allowed in this report, there are two ways to specify the column's contents. If the column merely contains data for a time period, the format is:

Cn = a

where n equals the number of the column and a is an element of (A, B, C...). 'A' represents the first time period requested, 'B' represents the second, etc. If the column is computed from one or more time periods, the following format is used:

Cn = f(a,b)

where f(a,b) is an arithmetic calculation involving time periods ('A', 'B', etc.), columns ('C1', 'C2', etc.), and/or numerical constants (which must always be preceded by an '!').

3. RDTITL is called, to read the headings for each column. RDTITL will read the next n lines (where n = the number of columns), treating each line as the next column heading. If an end-of-file is reached before RDTITL has read a heading for each column, the remaining columns will be output untitled by NEWPAG, and an error message will be output to device #15.

4. GETLST is now called to read the row specifications. It will continue reading the lines in the input file until a line beginning with '0999' is reached, or the end-of-file, or more than 50 accounts and account computations have been read. In the latter case, an error message will be output to device #15 and the MINIGAP run will stop. Otherwise, GETLST will store the accounts and computations using the arrays INDEX and EXP. The arry INDEX will contain a code to specify the row. If an account is requested, INDEX will be assigned a number relative to the account. (Since there are n accounts, these numbers will range from 1 to n.) If a calculation is desired, the number assigned will be greater than n, but less than 900. Format requests, such as paging and underlining, will receive codes between 900 and 910.

5. KEY is is set equal to the relative location of the next "activity" requested, using NXTKEY. If KEY=0, there are no more "activities" to be output, and control is returned to FPDREP.

5a. Otherwise, KFETCH is called to test the validity of the "activity". If the "activity" is valid (contained in the driver file), IXACTY is set to equal the index to the "activity". This is later used to access the name of the "activity". If KFETCH could not fetch the "activity" a new "activity" will be requested using NXTKEY (step #5).

6. For each time period requested, EXTRCT is called to retrieve the data from the master file and put it into the array DATA.

6a. If SPREAD data is requested for a time period (as specified in the input read by GETDAT) SPREAD is called to spread the data for that time period as desired (Since SPREAD keys on account codes, the development of these should be coordinated with SPREAD).

7. NEWPAG is now called to write the report header and the column headings for the page.

8. The elements of the array INDEX are now checked. If INDEX is greater than 900, the row contains no data, and control skips to WRTROW (step #11). If INDEX is less than or equal to the number of accounts, no row calculations have to be performed. Control skips over the computations of the rows to the second PARSE call (step #9b).

Otherwise, INDEX indicates that a row calculation (between accounts) has to be performed. For each account, and each column, PARSE is called to load the calculated data into the array A.

9a. PARSE is now called again, for each column, to do the column calculations (between time periods). This call, which is the second call to PARSE, is applied to those rows involving row calculations. The row labels are loaded into INAME at this time, using the user-supplied strings previously input. Control is sent to TABS (step #10).

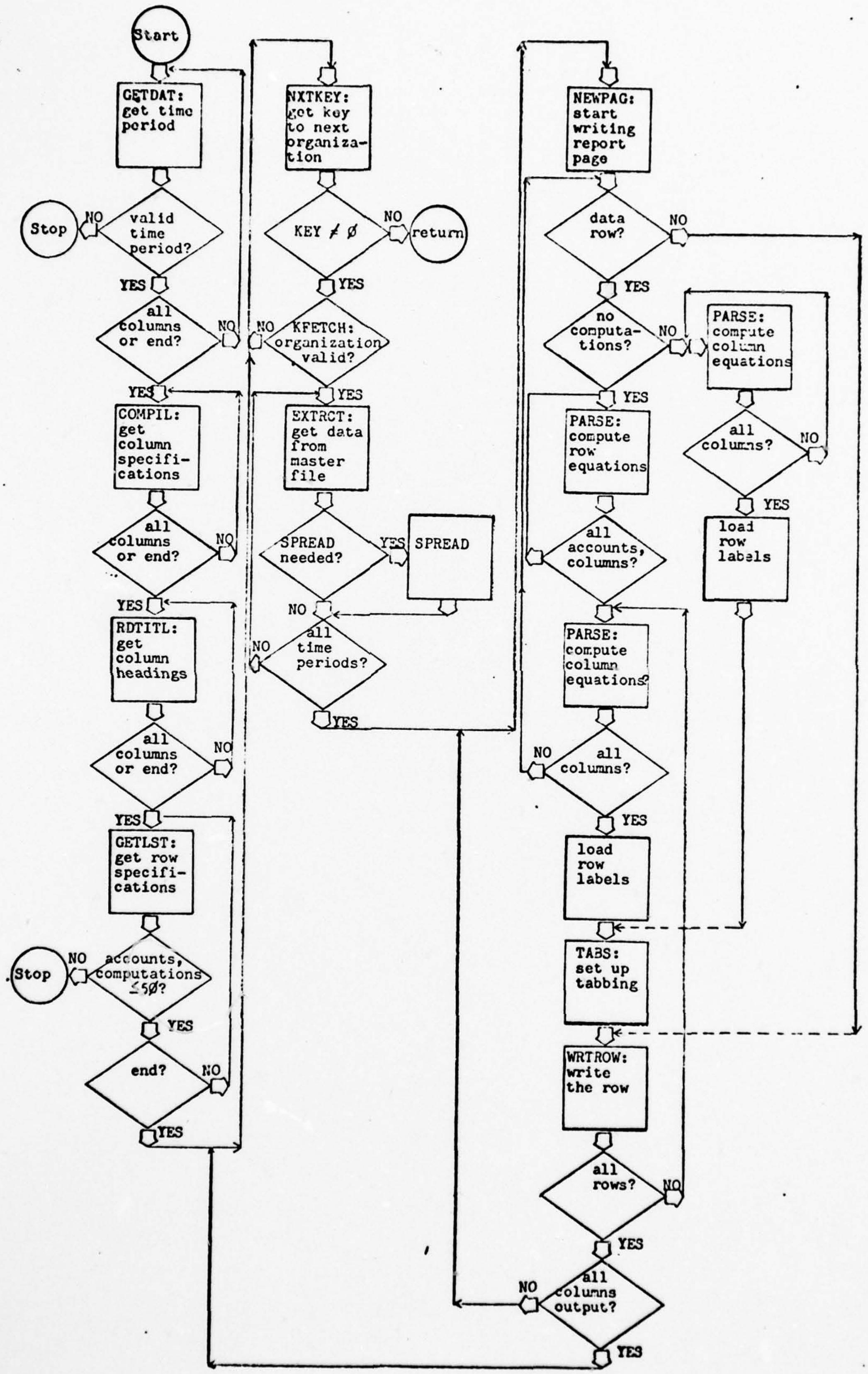
9b. The next call to PARSE (the third call) does the same thing (column calculations) as the prior call. However, this call applies only to non-computed account rows, while the previous call applies only to calculated rows. The row labels are loaded into INAME at this time, from the array MNAMES, which contains the account names.

10. TABS is now called, to provide horizontal tabbing for the INAME array, and store it in the JNAME array. Blank lines will be output to the report at this time for the vertical tabbing.

11. WRTROW is now called to write the row. The report generator will check to see if all rows have been written. If not, control skips back to check the contents of the new row (step #8). If all the rows have been output, REP005 now checks to see whether any columns did not fit on the 132 character-wide page. If there are any remaining columns, control skips to start a new page (step #7).

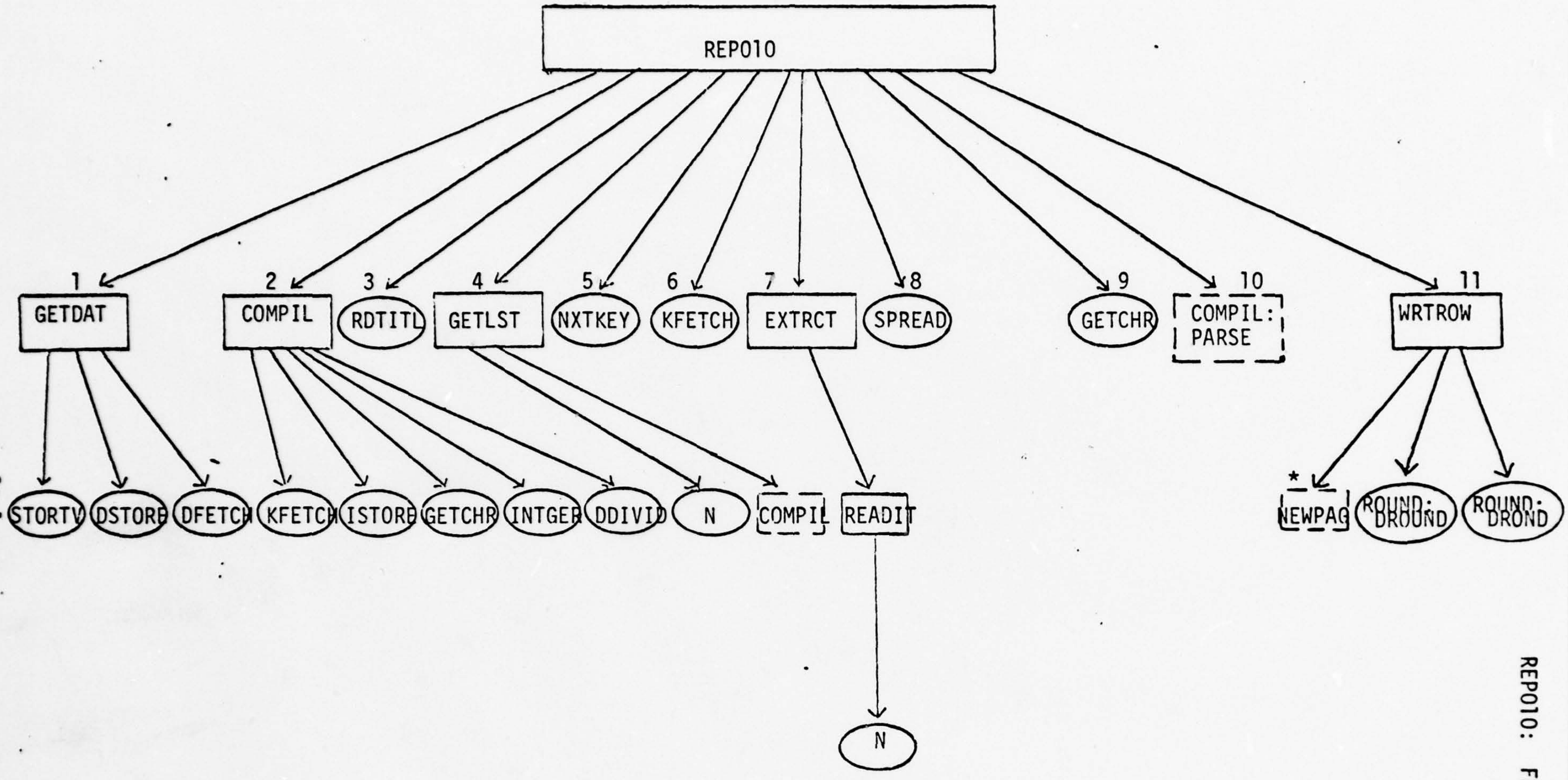
12. REP005 continues with a new page for the next activity, passing control to NXTKEY (step #5).

FLOW OF RÉPØØ5



MINIGAP

REPO10: FLOW OF CONTROL



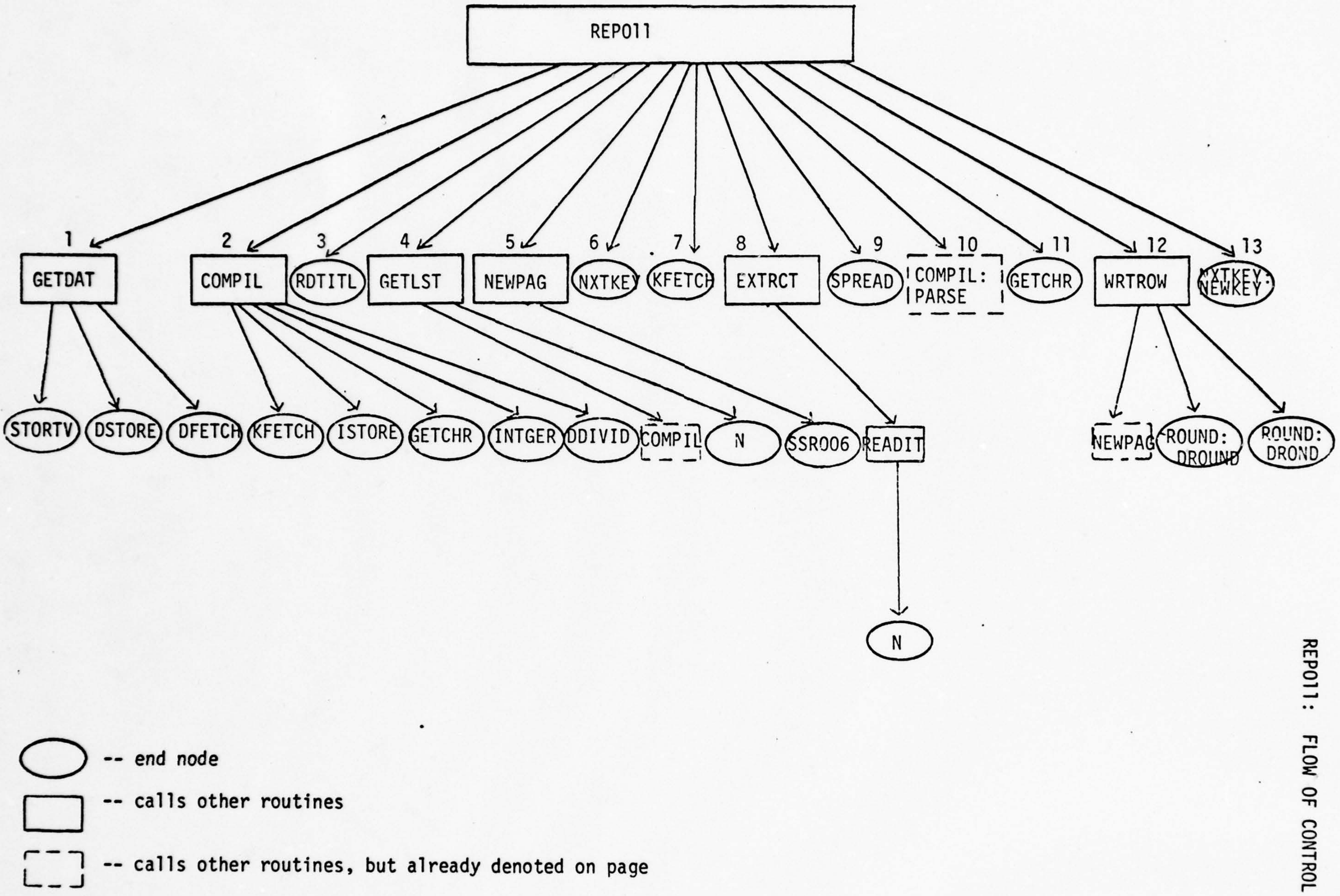
* Not referenced (due to: 6th parameter of call to WRTROW set to zero)

○ -- end node

□ -- calls other routines

⬚ -- calls other routines, but already denoted on page

REPO11: FLOW OF CONTROL



REP011

1. When REP011 is given control, TSIZE is set equal to the number of accounts involved (according to the driver file). Then, GETDAT is called to read the next lines which specify the time periods for each column. GETDAT will read a line for each column, until it reaches a line beginning with '0999', an end-of-file, or an invalid data type.

In the latter two cases, an error message is output to device #15, and the MINIGAP run stops. Otherwise, GETDAT sets up the variables which will later be used by EXTRCT to reference the data in the master file. GETDAT will also output its translation of each of the time periods to device #15 (in month, year and data type).

2. COMPIL is now called for each column, to read the column specifications. It will read one line for each column, unless it reaches an end-of-file. If there are too few column specifications, the remaining columns will be undefined. No error message will be output at this time; however, when RDTITL is called, later, and receives an end-of-file, an error message will be output concerning the column headings. The column specifications for this report have the same format as those for REP005. (See step #2 of REP005 for the formatting and contents of column specifications.)

3. RDTITL is called, to read the headings for each column. RDTITL will read the next n lines (where n = the number of columns), treating each line as a column heading. If an end-of-file is reached before RDTITL has read a heading for each column, the remaining columns will be output untitled by NEWPAG, and an error message will be output to device #15.

4. GETLST is now called to read the remaining lines. These lines should contain account codes or account computations. Each of these will be used for a different report; therefore, formatting specifications should not be used, since they have no meaning in this context. GETLST will continue reading lines, until it reaches a line beginning with '0999', an end-of-file, or more than 50 accounts and/or calculations have been specified.

In the latter case, an error message will be output to device #15, and the MINIGAP run will stop. Otherwise, GETLST will store the accounts and computations using the arrays INDEX and EXP. The array INDEX will contain a code to specify the report. If an account is requested, INDEX will be assigned a number relative to the account. (Since there are n accounts, these numbers will range from 1 to n.)

If a calculation is desired, the number assigned will be greater than n and less than 900. No formatting should be specified; however, this will not cause any errors at this point.

5. INDEX is now checkeu, to determine the contents of the report. If a formatting line is specified, control is given to WRTROW (step #12), with IXACCT and IXACTY retaining their previous, if any, values. (This should probably generate an error message of some sort, since the result is meaningless.) If an account is specified, the account name is loaded into INAME (to be used as the report label). For an account calculation, the label for the calculation is loaded from NAMES into INAME.

6. NEWPAG is now called to output the report header, and the column headings.

7. KEY is set equal to the relative location of the next organization requested, using NXTKEY. If KEY ≠ 0 (more organizations for report), control is sent to KFETCH (step #8).

7a. For KEY = 0 (no more organizations for this report), NEWKEY is called, to reinitialize the NXTKEY variables. If there are any more reports to be output (as specified by the input to GETLST) control now returns to check INDEX (step #5). Otherwise, control is returned to FPDREP (main).

8. KFETCH is now called to check the validity of KEY, and to store the index to the organization in IXACTY. If the KEY is invalid, the next KEY is fetched (returning control to step #7).

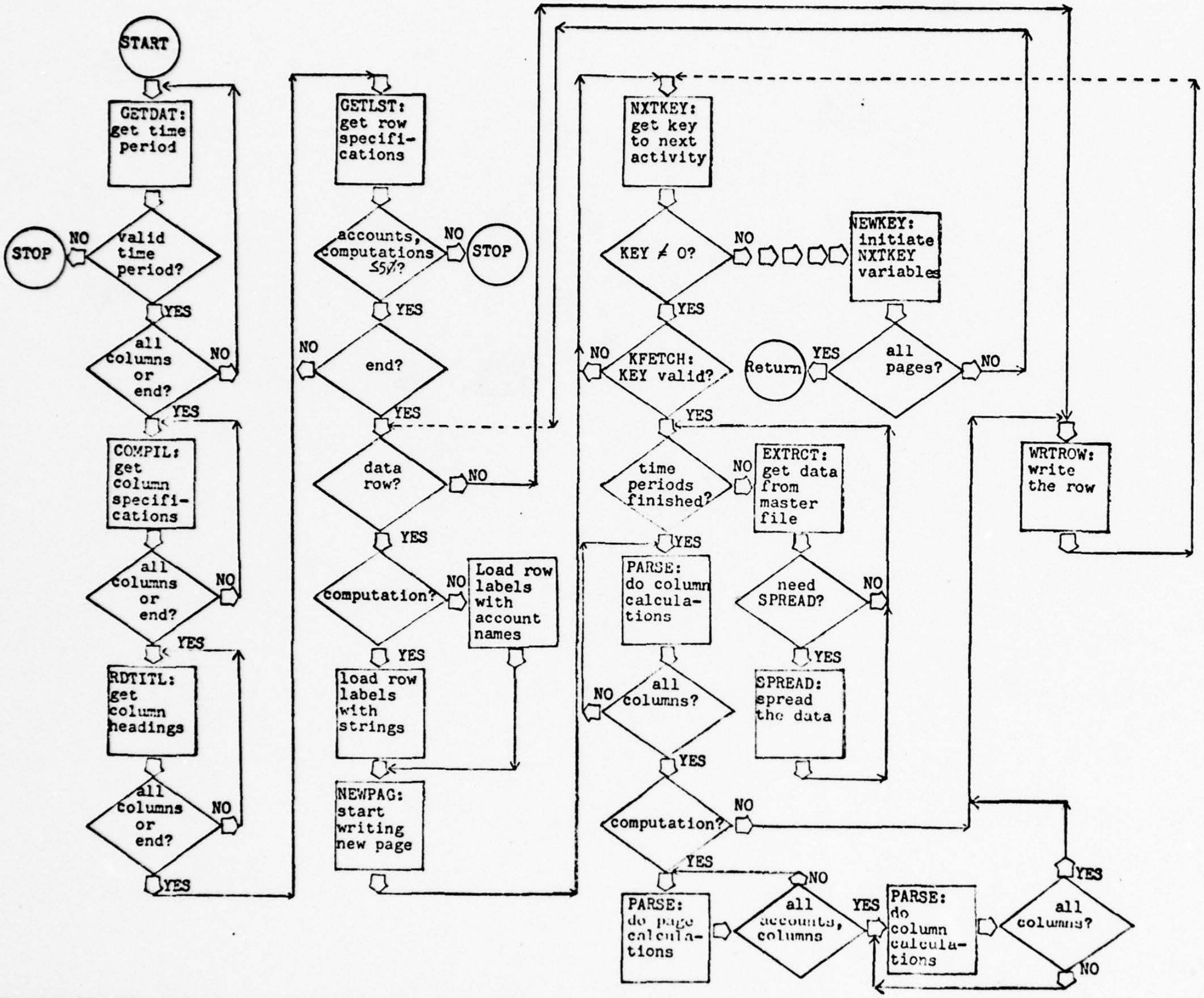
9. For each time period, EXTRCT is called to load the data from the master file into the array DATA. If the time period has specified that the data be spread, SPREAD is also called. It will manipulate the data in the array DATA to appear as thought it were spread over a period of time.

10. PARSE is now called for each column, to load the proper data into the correct column, and perform any column calculations as specified. After this is completed, the report generator will check to see if a computation has been specified for the report. If not, control skips to WRTROW (step #12).

11. Since a page computation has been requested, PARSE is called, for each account and column, to compute the data as specified. Then, for each column, PARSE is called, again, so that all relations between columns will be those specified (as input to COMPIL) by the user.

12. WRTROW is now called to write the row for this organization. It uses the IXACTY index to write the activity name for the row label. If the data to be output represents a subtotal, IFLAG is set to cause the row to be set off by hashed lines. The next organization is fetched after this row is written. (Control returns to step #7.)

FLOW OF REPØ11



SECTION 4. MINIGAP MODULE CHARTS

Software Name: BLOCKB
Type: Common Block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/9/78

Purpose of Routine

This common block is loaded from the driver file, using FPDØØ1. Three levels of organization are permitted, i, j, and k. An organization/activity is uniquely identified by i, j. F (i, j) contains the relative record location of the organization i, j within the master file. Each jn is also associated with a km. G(jn) equals km. For example: The organizations (1,2), (2,2), (3,2), and (4,2), are all subtotaled into (2Ø,2), and are all grouped into one group at the second level. This one group at the second level is a member of another group, K at the third level. If group 3 at the third level contains the groups 2,5, and 6 at the second level, then the following will result:

G (1)=
G (2)= 3
G (3)=
G (4)=
G (5)= 3
G (6)= 3

etc.

G() & F() are both set up by FPDØØ1, using the driver file. The function NXTKEY uses these arrays to sequentially access desired records within a range. For example, if the range given is: i=Ø, j=Ø, k=3 then, using the above example, NXTKEY will first access the record at relative location F(1,2). (con't on next page)

Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
N/A	N/A	N/A	FPDØØ1 NXTKEY (function)	N/A	N/A	N/A

Error Codes/Messages Line #	Generated Message	Arguments Name	Arguments Function	Commons Referenced	Comments
N/A		F	contains relative record location of each activity integer *2 array, length= 2ØX2Ø	N/A	see NXTKEY
		G	contains a third level grouping for each column (2nd level) integer *2 array, length= 2Ø		

Software Name: BLOCKB
Type: Common Block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
Succeeding calls to NXTKEY will result in: F(2,2), F(3,2), F(4,2)...F(2Ø,2), F(i,5)...F(2Ø, 6),Ø The zero at the end indicates there are no more organizations/activities within the desired range.							

Error Codes/Messages Generated Line #	Message	Arguments Name	Function	Commons Referenced	Comments

Software Name: BLOCKD
Type: BLOCK Data
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 9/20/77.

Purpose of Routine	Entry Points	Routines Called Subroutine	 (Entry)	Called By Name	 (Entry)	Functions Accessed	Files Referenced
Initializes the elements of the Tl,SPACE, MINVAL, MAXVAL, IDEFLT, TITLE and UNITS arrays. (see documentation) for: WORKA WORKB PARAMS TITLES	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Error Codes/Messages Generated Line #	 Message	Arguments Name	 Function	Commons Referenced	Comments
N/A		N/A		WORKA WORKB PARAMS TITLES	

Software Name: BLOCKD
Type: Common block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 11/11/77

Purpose of Routine

This common block contains the short and long name of each organization in the driver file, and the tables necessary to use the STORTV routines. These routines allow quick access of an index which can also be used for other information about an organization (i.e., the UIC code), without searching.

FPDØØ1 reads the driver file and makes appropriate calls to the STORTV routines to set up this block.

The report generators for types 5, 1Ø and 11 use the information contained in this common block.

Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
N/A	N/A		FPDØØ1 REPØØ5 REPØ1Ø REPØ11		N/A	N/A

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
N/A		D1	contains 8 character organization names double precision array, length = 16Ø	N/A	see: STORTV ISTORE/DSTORE KFETCH/DFETCH
		D2	used as TABLE 2 for STORTV routines integer *2 array, length = 16Ø		
		(con't on next			

Software Name: BLOCKD
Type: Common block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 11/11/77

Con't

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
		D3 integer *2 array, length = 16Ø	used as TABLE 3 for STORTV routines		
		DNAMES Real array, length = 11 X 16Ø	contains 44 charac-ter organi-zation names		
		DFREE (con't on next page)	used as		

Software Name: BLOCKD
Type: Common block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 11/11/77

Con't

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
		integer * 2	FREE for STORTV routines		

Software Name: BLOCKK
Type: Common block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/11/78

Purpose of Routine	Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
This common block contains the relative location of the record for each organization, and the tables necessary to use the STORTV routines. FPDØØ1 reads the driver file and loads BLOCKK using the STORTV routines (see BLOCKD, purpose). In the master file, there is a block of data for each time period. The number contained in K1 gives the location within these blocks. For example: Suppose the number contained in K1 for organization A equals 1Ø3. Then, for the time period beginning at location 1ØØ, the data for organization A is located at 2Ø2. For the time period beginning at 4Ø5, the data is at 5Ø7. (4Ø5 + 1Ø3 -1) This information is used in each report generator.	N/A	N/A	N/A	FPDØØ1 REPØØ2 REPØØ4 REPØØ5 REPØ1Ø REPØ11	N/A	N/A	N/A

Error Codes/Messages Line #	Generated Message	Arguments Name	Arguments Function	Commons Referenced	Comments
		K1	contains relative record location for each organization integer *4 array, length=16Ø		see: BLOCKD, purpose STORTV ISTORE/DSTORE KFETCH/DFETCH .
		K2	used as TABLE 2 for STORTV routines integer *2 array length=16Ø		
		K3	used as TABLE 3 for STORTV routines integer *2 array, length=16Ø		

(cont' on next page)

Software Name: BLOCKK
Type: Common Block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/11/78

(con't)

Purpose of Routine	Entry Points	Routines Called		Called By		Functions Accessed	Files Referenced
		Subroutine	(Entry)	Name	(Entry)		

Error Codes/Messages	Generated	Arguments		Commons	
Line #	Message	Name	Function	Referenced	Comments
		KFREE	used as FREE for STORTV routines		
		integer *2			

Software Name: BLOCKM
Type: Common Block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/10/78

Purpose of Routine

This common block contains the account codes and the names of each account. FPDØØ1 reads the driver file and sets up these tables using the STORTV routines. The function N essentially performs a FETCH for these tables, but also keeps a record of invalid fetches. INITLZ stores the account codes (M1) in the T1 array.

SPREAD is used to spread out the data over several periods. It is a specialized routine, in that it keys on the account code and does different things with different accounts.

Development of any account code should be coordinated with SPREAD modifications.

Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
N/A	N/A		FPDØØ1 INITLZ SPREAD N (function) REPØØ2 REPØØ4 REPØØ5 REPØ1Ø REPØ11	N/A	N/A	N/A

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
N/A		M1	contains the 4 character account codes integer *4 arrary, length=348	N/A	see: STORTV ISTORE/DSTORE SPREAD KFETCH/DFETCH
		M2	used as TABLE 2 for STORTV routines integer *2 array, length=348		
		M3	used as TABLE 3 for STORTV routines integer *2 array, length=348		
		con't on next page			

Software Name: BLOCKM
Type: Common Block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/10/78

(con't)

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Generated Line #	Message	Arguments Name	Function	Commons Referenced	Comments
		MNAMES	contains the 36 character account names		
		integer	*4 array length= 9 x 348		
		MFREE	used as FREE for STORTV routines		
		integer	*2		

Software Name: BLOCKT
Type: Common Block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 11/11/77

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
This common block is set up by GETLST, from the user input list. Tabbing information is stored in HTABS & VTABS (for HTABS=3, tab 12 spaces, HTABS=2, tab 8 spaces, etc). Since the dimension of INDEX is 348, no more than 348 lines are allowed in the content of the report (exclusive of heading information and blank lines). If INDEX is in the 9ØØ - 91Ø range, combinations of data, hyphens, & equals are output. If INDEX is in the SIZE +1 thru 899 range, a computation is output, and if INDEX is in the Ø - SIZE range, the data for that account is output. (SIZE = number of accounts)	N/A	N/A		GETLST REPØØ2 REPØØ5 REPØ11	N/A	N/A	N/A

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
N/A		INDEX	contains the index to:(1) the desired account in M1 (2) the desired computation in SPACE/T1 (3) Code for hyphens equals, pages integer *2 array, length=248	N/A	see: WRTROW, purpose, INDX

(con't on next page)

Software Name: BLOCKT
Type: Common block (con't)
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 11/11/77

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Generated Line #	Message	Arguments Name	Function	Commons Referenced	Comments
		HTABS	contains the number of 4 character spaces to tab horizontally		
		integer*	2 array, length=348		
		VTABS	contains the number of blank lines to skip		
		integer	*2 array, length=348		

Software Name: BLOCKU
Type: Common block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 11/11/77

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
This common block contains the UIC s of each organization and the tables necessary for using the STORTV routines. FPDØØ1 reads the driver file and loads this block using the STORTV routines. Presently, the information in this common block is not used, and this block could be dispensed with by modifying FPDØØ1.	N/A	N/A		FPDØØ1		N/A	N/A

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
N/A		U1	contains the unique identifier codes (UIC) double precision array, length=16Ø	N/A	
		U2	Used as TABLE 2 in STORTV routines integer *2 array, length=16Ø		
		U3	used as TABLE 3 in STORTV routines integer *2 array, length=16Ø		
		(con't on next page)			

Software Name: BLOCKU
Type: Common block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello (con't)
Date Last Revised: 11/11/77

Purpose of Routine	Entry Points	Routines Called Subroutine	 (Entry)	Called By Name	 (Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	 Function	Commons Referenced	Comments
		UFREE integer *2	Used as FREE in STORTV routines		

Software Name: COMPIL
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenançe: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine	Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	Called By (Entry)	Functions Accessed	Files Referenced
Together, COMPIL and PARSE serve to read the input parameters and expressions, line by line, and translate for use by MINIGAP. COMPIL: reads the input string, sets up ISENT which contains the indices into T1 and SPACE for each expression. PARSE: Computes expressions, etc., using the ISENT (ADDRSS) array and a STACK format.	PARSE	STORTV	KFETCH ISTORE	FPDREP	COMPIL PARSE	GETCHR DDIVID INTGER	INPUT (defaults to #1)
				GETLST	COMPIL		
				RDPARM	COMPIL PARSE		#15
				REPØØ2	PARSE		
				REPØØ4	COMPIL PARSE		
				REPØØ5	COMPIL PARSE		
				REPØ1Ø	COMPIL PARSE		
				REPØ11	COMPIL PARSE		

Error Codes/Messages Generated Line #	Message	Arguments Name	Arguments Function	Commons Referenced	Comments
8Ø1	*****COMPIL*****...	RETN	Set to TRUE if EOF reached during READ logical	WORKA WORKB	FLAG-should eliminate useage, always output the input string to #15
	ERRORS				
9Ø1	**COMPIL**ILLEGAL CHARACTER IN NUMERIC STRING	INPUT	# of input device integer *4		ISENT and ADDRSS Should be combined, since COMPIL outputs ISENT for use in PARSE as ADDRSS. No need for the duplication.
9Ø2	**COMPIL**CANNOT STORE AAAA IN SYMBOL TABLE	FLAG	If set to TRUE, writes to #15 what was READ from INPUT. logical		
9Ø3	**COMPIL**COMMAND STRING>71 CHARACTERS	ISENT	Will contain		
9Ø4	**PARSE**STACK OVERFLOW				
9Ø5	**PARSE**SYNTAX ERROR				

(con't on next page)

Software Name: COMPIL
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenançe: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
			numeric code equal to the decoded character's position in T1 integer *2 array, length = 8Ø		
		STRING	will contain the row label for a computed expression. integer *4 array, length = 18		
		ADDRSS	same as		

CONT. ON NEXT PAGE

Software Name: COMPIL
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenançe: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Generated		Arguments		Commons	
Line #	Message	Name	Function	Referenced	Comments
			ISENT, but used in PARSE		
		integer *2 array, length = 8Ø			
		R3	Returns numerical code from SPACE associated with the given character or expression.		
		double precision			

Software Name: DDIVID
Type: Function
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 11/11/77

Purpose of Routine	Entry Points	Routines Called Subroutine	 (Entry)	Called By Name	 (Entry)	Functions Accessed	Files Referenced
Returns the result of a double precision division. If either the numerator or denominator is approximately Ø, (less than .ØØØØØØ1) a zero is returned.	N/A	N/A		COMPIL	PARSE	N/A	N/A

Error Codes/Messages Line #	Generated Message	Arguments Name	 Function	Commons Referenced	Comments
N/A		DDVDND	numerator double precision	N/A	
		DDVISR	denominator double precision		

Software Name: EXTRCT
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine	Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	Called By (Entry)	Functions Accessed	Files Referenced
Used to compute the correct record location (depending upon data type) to be read using READIT. Will also read the proper records & perform subtraction to produce non-cumulative data. (Is somewhat data dependent in that it will only subtract data if the numbering of the respective account implies subtraction should take place.) For example, if the account code is: 1XXX, 2XXX, or 7XXX no subtraction is performed. This may be changed by changing the computed go-to's.	N/A	READIT	N/A	REPØØ2 REPØØ4 REPØØ5 REPØ11	N/A	INTGER	see comments

Error Codes/Messages Generated Line #	Message	Arguments Name	Function	Commons Referenced	Comments
None; Returns the variable 'RETN' set to 'TRUE' and 'TEMP1()' set to zero if error encountered. Possible errors: 1. The appropriate variable (of I,J,K,L or M) = Ø 2. READIT returns error code.		RETN logical	Returned as true in case of error	BLOCKM	EXTRCT calls READIT, which reads the MASTER file, set to logical unit #2
		KEY integer *4	Relative location of activity record		
		ITYPE integer *2	Numerical representation of data type		
		I	Starting location of block		

(con't on next page)

Software Name: EXTRCT
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called		Called By		Functions Accessed	Files Referenced
		Subroutine	(Entry)	Name	(Entry)		

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
		integer *2	for requested data period		
		J integer *2	Starting location of block for previous month		
		K integer *2	Starting location of block for previous quarter		
		L	Starting location		

(con't on next page)

Software Name: EXTRCT
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
			of block for previous year at this time		
		integer *2			
		M	Starting location of block for base year at this time		
		integer *2			
		JJ	Equals the division for the requested rounding		
		integer *4			
		DNAME	activity		

cont. on NEXT page
(cont. on next page)

Software Name: EXTRCT
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenançe: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
			short name		
		double precision			
		MM	number of month (2-digit)		
		integer *4			
		UIC	Code of activity		
		double precision			
		TEMP1	Returns data for all accounts for desired activity		
		double precision array, length= # of accounts			
		SIZE	Number of accounts		
		integer *4			

Software Name: FPDØØ1
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine	Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	Called By (Entry)	Functions Accessed	Files Referenced
Reads data from driver file and puts it in tables in the common blocks, using the necessary STORTV routines. It is used only once, and is the first routine called.	N/A	STORTV	STORTV DSTORE ISTORE KFETCH	FPDREP	N/A	N/A	#15 - output file for diagnostic-type guides IUNIT (#7) - DRIVER.nnn (DRIVER file)

Error Codes/Messages Generated Line #	Error Codes/Messages Generated Message	Arguments Name	Arguments Function	Commons Referenced	Comments
ONE ERROR MESSAGE: 9Ø1	**FPDØØ1**CANNOT STORE AAAAAAAA IN SYMBOL TABLE	PRINT logical	if true, causes all generated messages to be output to #15.	BLOCKB BLOCKD BLOCKK BLOCKM BLOCKU	Formerly, FPDØØ1 read the first record of the master file and stored it in tables of common BLOCKZ. Due to problems w/common and equivalences, this read was moved to GETDAT and BLOCKZ eliminated. This results in duplicate reading and storing of the first record every time
8Ø1	**FPDØØ1** SYMBOL TABLE STORAGE BEGUN				
8Ø2	**FPDØØ1** NNN ACTIVITY NAMES, UICS, KEYS				
8Ø5	**FPDØØ1** NNN ACCOUNT SYMBOLS, NAMES	ISIZE	length of records in master file; used to determine the # of accounts & the #		
82Ø	**PFDØØ1** SYMBOL TABLE STORAGE COMPLETE				

(con't on next page)

(con't on next page)

Software Name: FPDØØ1
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
			of data types.		GETDAT is called.
		Integer *4			

Software Name: FPDREP
Type: Main
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
Calls FPDØØ1 to set up symbol tables. Uses IFETCH (system subroutine) to fetch in overlays which contain respectively: INITLZ REPØØ2 REPØØ4 REPØØ5 REPØ11	N/A	FPDØØ1 IFETCH INITLZ COMPIL NODATA RDPARM REPØØ2 REPØØ4 REPØØ5 REPØ11	 COMPIL PARSE	N/A	N/A	N/A	#15

Calls INITLZ to set up symbol tables for WORKA and BLOCKM.
Calls COMPIL & PARSE to determine the report type.
Calls RDPARM to read input parameters.
Calls NODATA to initialize its use.
Calls desired report as determined through COMPIL and PARSE.
Loops through all reports requested.

Error Codes/Messages Generated Line #		Message	Arguments Name	Function	Commons Referenced	Comments
ONE ERROR MESSAGE:			N/A		N/A	The array 'TITLE' was previously used ~~us~~ as input to HEADER, which was to write a header on the report. At present, HEADER is not used.
9Ø2	**FPDREP**	REPORT TYPE NNNNN NOT VALID				
8Ø1	**FPDREP**	REPORT #NNNNN, REPORT TYPE NNN.				
9Ø1	**FPDREP**	END OF REPORT PROCESSING				

Software Name: GETCHR
Type: Function
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 11/11/77

Purpose of Routine	Entry Points	Routines Called		Called By		Functions Accessed	Files Referenced
		Subroutine	(Entry)	Name	(Entry)		
Performs same function as KFETCH. Could easily replace this with: CALL KFETCH (SEND,IWORD, T1, T2, T3, 1Ø24, 1Ø23,GETCHR) IF(SEND) WRITE (15,8Ø1)IWORD Could also change KFETCH & DFETCH to set INDX equal to L if error. However, none of the above has been done, since it is easiest to leave it as is, with no large amount of space lost.	N/A	N/A		COMPIL REPØØ5 REPØØ2 REPØØ4 REPØ1Ø REPØ11	COMPIL	N/A	#15

Error Codes/Messages Generated		Arguments		Commons Referenced	Comments
Line #	Message	Name	Function		
8Ø1	**GETCHR**CANNOT FIND AAAA IN SYMBOL TABLE	IWORD integer *4	"Word" to be fetched from T1	WORKA	see: KFETCH

Software Name: GETDAT
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine	Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	Called By (Entry)	Functions Accessed	Files Referenced
Reads first record of master file to get datatype names and locations of the data blocks. The portion of the input parameter list which deals with datatypes is read, and the proper year & month is computed and output. A numeric value is assigned to ITYPE to be used in EXTRCT to determine what type of data is required and what must be done to obtain it.	N/A	STORTV	STORTV DSTORE DFETCH	REPØØ2 REPØØ4 REPØØ5 REPØ11	N/A	N/A	#2 (MASTER.XXX)

Error Codes/Messages Generated Line #		Message	Arguments Name	Arguments Function	Commons Referenced	Comments
8Ø3	**GETDAT**	DATATYPE YEAR MONTH PERIOD SPREAD	RETN logical	set to 'TRUE' if 'END' is reached before NUM reads	PARAMS	The read of the first record of the master file was formerly located in FPDØØ1; at its present location, it is inefficient, since the same read is done everytime GETDAT is called.
8Ø4	**GETDAT**	AAAAAAAA AAAA AAAA A A				
ERROR MESSAGES:						
9Ø1	**GETDAT**	UNEXPECTED END OF FILE	NUM integer *4	# of datatypes to be read		
9Ø2	**GETDAT**	AAAAAAAA NOT FOUND	L integer *4	# of datatypes actually read		
9Ø3	**GETDAT**	CANNOT STORE AAAAAAAA IN SYMBOL TABLE	ILOC	starting location of respective		

(con't on next page)

Software Name: GETDAT
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called		Called By		Functions Accessed	Files Referenced
		Subroutine	(Entry)	Name	(Entry)		

Error Codes/Messages Generated Line #	Message	Arguments Name	Function	Commons Referenced	Comments
			datatype integer *2 array, Length= NUM		
		IPM	index to previous month datatype integer *2 array, length= NUM		
		IPQ	index to previous quarter datatype integer *2 array, length= NUM		
		IPY	index to previous		

(con't on next page)

Software Name: GETDAT
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called		Called By		Functions Accessed	Files Referenced
		Subroutine	(Entry)	Name	(Entry)		

Error Codes/Messages Generated		Arguments		Commons Referenced	Comments
Line #	Message	Name	Function		
			year at this time datatype integer *2 array, length= NUM		
		IPP	index to base year at this time datatype integer *2 array, length= NUM		
		ITYPE	numeric value to identify "type" of data		

(con't on next page)

Software Name: **GETDAT**
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	Called By (Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Generated Line #	Error Codes/Messages Generated Message	Arguments Name	Arguments Function	Commons Referenced	Comments
		integer *2 array, length= NUM			
		MONTHS	Alpha representation of month of desired datatype		
		integer *4 array, length= NUM			
		IYEARS	alpha representation of year of desired datatype		

(con't on next page)

Software Name: GETDAT
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
		integer *4 array, length= NUM			
		SIZE	length of records in master file; used to compute number of datatypes		
		integer *4			

Software Name: GETLST
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 9/21/77

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
Reads the portion of the input string which deals with the accounts & account computations. Sets up INDEX with the number of the account location in SPACE (the # of accounts), or a 900 code for hyphens, equals, or paging, or a number between (the # of accounts) and 900, for the computation. The number used for INDEX is either obtained from the function N, from GETLST itself, or from the EXP array in COMPIL. HTABS and VTABS are set up according to the values read from the input string.	N/A	COMPIL	COMPIL	REP∅∅2 REP∅∅5 REP∅11	N/A	N	INPUT (1) #15

	Error Codes/Messages Generated Line #	Message	Arguments Name	Function	Commons Referenced	Comments
9∅1	**GETLST**	TOO MANY CALCULABLE EXPRESSIONS	NCOUNT integer *4	# indices returned (output)	BLOCKT PARAMS	see: BLOCKT, purpose WRTROW, purpose
			ICOUNT integer *4	expressions to be calculated (1≤ ICOUNT≤ 15) output		
			KCOUNT integer *4	input-# of columns ≤ 15		
			EXP	the expressions		

(con't on next page)

Software Name: GETLST
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 9/21/77

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine (Entry)	Called By Name (Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Generated Line #	Message	Arguments Name	Function	Commons Referenced	Comments
			to be computed by PARSE		
		integer *2 array, 8Ø X 15			
		NAMES	array of labels for the expressions to be calculated.		
		integer *4 array, 18 X 5Ø			

Software Name: INITLZ
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 9/21/77

Purpose of Routine	Entry Points	Routines Called		Called By		Functions Accessed	Files Referenced
		Subroutine	(Entry)	Name	(Entry)		
Stores the elements of T1 into T1, using the STORTV routines for future searches. Stores the elements of M1(the account codes) into T1 using the same process.	N/A	STORTV	STORTV ISTORE	FPDREP	N/A	N/A	#15

Error Codes/Messages Generated		Arguments		Commons Referenced	Comments
Line #	Message	Name	Function		
901	*** SYMBOL TABLE FULL, CANNOT STORE *** AAAA	N/A		WORKA BLOCKM	

Software Name: INTGER
Type: General Function
Software Author: S. Masiello
Person in charge of maintenance: S. Masiello
Date Last Revised: 11/11/77

Purpose of Routine

This function returns an integer, given an alphanumeric string of consecutive digits contained in an array ALFA.

EXAMPLE:
GIVEN:
ALFA (1)='ON T'
ALFA (2)='HE 1'
ALFA (3)='1TH'
IERR =Ø
NUMBER=INTGER (ALFA,8,2,IERR)

RESULT:
ALFA remains unchanged
IERR=Ø
NUMBER =11

A plus sign has no effect, and a minus sign causes a negative number to be returned. Signs are only expected at the beginning of the number. Blanks are treated as zeros.

Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	Called By (Entry)	Functions Accessed	Files Referenced
N/A	N/A		SPREAD EXTRCT COMPIL	 COMPIL	N/A	N/A

Error Codes/Messages Line #	Generated Message	Arguments Name	Arguments Function	Commons Referenced	Comments
N/A		ALFA real array,length 1ØØ	contains alpha representation of a number	N/A	Any nonnumeric character will cause an error return. INTGER will equal the number decoded before the error & IERR=1 A sign (+ or -) at the ISTART position, will not cause an error. (con't on next page)
		ISTART integer *4	location of the left-most digit of the number in ALFA		
		(Con't on next page)			

Software Name: INTGER
Type: General Function
Software Author: S. Masiello
Person in charge of maintenance: S. Masiello (con't)
Date Last Revised: 2-9-78

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
		IWIDTH	length of number in ALFA		Example: Given: ALFA(1)='123 ' ALFA(2)='4T56' IERR=Ø NUMBER=INTGER Result: ALFA remains unchanged NUMBER=123Ø4 IERR=1
		integer	*4		
		IERR	set to 1 if error, otherwise unchanged		
		integer	*4		

Software Name: ISTORE,DSTORE
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine

These two routines are used to store integer items into an integer array (ISTORE) and to store double precision items into a double precision array (DSTORE), using the following hashing formula: HASH=MOD(ITEM,ID)+1 STORTV must be used before ISTORE or DSTORE, to initialize the tables. Synonyms are chained using TABLE2 - the last synonym stored is the first one fetched. To avoid long synonym chains, ID should be made as large as possible (L-1). To prevent storing duplicate items, a call to KFETCH/DFETCH could be made first; a normal return would indicate that the item has already been stored. If @ TABLE is full, RETN will be set TRUE and the item will not be stored.

NOTE:
@ = I or D, respective to
ISTORE (integer) or
DSTORE (double precision)

Entry Points	Routines Called Subroutine	 (Entry)	Called By Name	 (Entry)	Functions Accessed	Files Referenced
N/A	N/A		FPDØØ1	DSTORE	N/A	N/A
				ISTORE		
			GETDAT	DSTORE		
			INITLZ	ISTORE		
			COMPIL	ISTORE		

Error Codes/Messages Line #	Generated Message	Arguments Name	 Function	Commons Referenced	Comments
N/A		RETN	set to zero if FREE=0 (no free space)	N/A	Note: These routines are often referred to as entry points of STORTV. They are separate subroutines, although they could be incorporated into STORTV as entry points.
		logical			
		ITEM/DWRD	item to be stored in @ TABLE		
		integer *4/ double precision			
		@TABLE	table in which items are stored		See also: STORTV DFETCH/KFETCH

con't on next page

Software Name: ISTORE,DSTORE
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
		integer *4/ double precision array, length=L			
		TABLE2	set to point to previous synonym of ITEM		
		integer *2 array, length=L			
		TABLE3	set to point to location of ITEM in @TABLE		
		integer *2 array, length=L			
		L	used to dimension		

(con't on next page)

Software Name: ISTORE,DSTORE
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
			the tables		
		integer *4			
		ID	divisor used in hashing function, normally =1-1		
		integer *4			
		FREE	set to point to next free space in @ TABLE		
		integer *2			

Software Name: KFETCH,DFETCH
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine

These two routines are both used to fetch ITEMS which have been stored using ISTORE or DSTORE. KFETCH fetches integer items from an integer array, which were stored using ISTORE. DFETCH fetches double precision items from a double precision array, which were stored using DSTORE. If an item is not found (the element in TABLE3 corresponding to the hash of the item set to zero) RETN is set to TRUE. Otherwise, INDEX is set to point to the ITEM in @TABLE.

See also: STORTV
ISTORE/DSTORE

Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
N/A	N/A		GETDAT	DFETCH	N/A	N/A
			COMPIL	KFETCH		
			REPØØ2	KFETCH		
			REPØØ4	KFETCH		
			REPØØ5	KFETCH		
			REPØ1Ø	KFETCH		
			REPØ11	KFETCH		

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
N/A		RETN logical	set to TRUE if unable to find ITEM in @TABLE	N/A	@ = I or D, for integer or double precision STORTV must be used to initialize the tables. @FETCH should not be used before this, as infinite looping could result. NOTE: These routines are often referred to as entry
		ITEM/DWRD integer *4/ double precision	word to be fetched from @TABLE		
		ITABLE/DTABLE integer *4/ double precision	array where the list is stored		

(Cont. on NEXT Page

Software Name: KFETCH,DFETCH
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
		array, length=L TABLE2	indices to @TABLE; set to zero if no synonym		points of STORTV. See also ISTORE,DSTORE: COMMENTS.
		integer *2 array, length=L TABLE3	contains the index to the last item stored for that hash code		
		integer *2 array, length=L			

(con't on next page)

Software Name: KFETCH,DFETCH
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called		Called By		Functions Accessed	Files Referenced
		Subroutine	(Entry)	Name	(Entry)		

Error Codes/Messages	Generated	Arguments		Commons	Comments
Line #	Message	Name	Function	Referenced	
		L integer *4	used to dimension the tables		
		ID integer *4	divisor used in hashing function		
		INDEX integer *4	Set to the position of the ITEM in @TABLE, if found; else, set to zero		

Software Name: N
Type: Function
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
Returns the location of 'ITEM' in 'M1'. If 'ITEM' cannot be found, an error message is issued and the value of 'N' is returned as zero (Ø).	N/A	N/A		GETLST READIT	N/A	N/A	#15

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
5Ø	EXECUTION HALTED, 1ØØØØ ILLEGAL ACCOUNTS	ITEM integer *4	the alpha representation of the desired item	BLOCKM	
		ISIZ integer *4	the number of accounts		

Software Name: NEWPAG
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
Outputs (to IOUTPT) the heading for each report, including the title of the report, the date, the column headings for the report, the page number, and the title used over the rows; everything that WRTROW doesn't do in writing a report.	N/A	SSRØØ6	N/A	WRTROW REPØØ2 REPØØ4 REPØØ5 REPØ11	N/A	N/A	IOUTPT (defaults to #1

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
N/A		STUB	Contains string to be used above the row labels integer *4 array, length = LENGTH	PARAMS TITLES	Previously, TITLE, HYPHN, STUB and OUTPUT were handled as logical *1 which would eliminate the need for the internal routines, ILBYTE & ISBYTE.
		LENGTH	dimension of STUB, ≤ 9 integer *4		
		ISTART	tells at what point in the TITLE array to		

(con't on next page)

Software Name: NEWPAG
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenançe: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Generated Line #	Message	Arguments Name	Function	Commons Referenced	Comments
			pick up the column titles		
		integer *4			
		ISTOP	not used; should be removed		
		integer *4			

Software Name: NODATA
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenançe: S. Masiello
Date Last Revised: 9/22/77

Purpose of Routine	Entry Points	Routines Called Subroutine	 (Entry)	Called By Name	 (Entry)	Functions Accessed	Files Referenced
Gives list of activities which were not included in report because there was no data available concerning them. Is also used to initialize the NXTKEY function by calling NEWKEY.	N/A	N/A	N/A	FPDREP REPØØ4	N/A	NXTKEY- at entry NEWKEY	IOUTPT (defaults to #15)

Error Codes/Messages Line #	Generated Message	Arguments Name	 Function	Commons Referenced	Comments
701	The following NN activities are not included in this report because data was not available.	I	Index to activity which had no data integer *4	PARAMS BLOCKD	Will only output a message if REPØØ4 receives error message from EXTRACT and ICOL≠2Ø. This EXTRCT error could be caused by not being able to determine the record location (I,J,K,L or M =Ø in EXTRCT & tested) or if the 8 character with the activity name are blank in the accessed record

So re : N EY
Type: Function
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Software Name: NXTKEY

Purpose of Routine

NXTKEY looks in the "F" array for the next Key to a record.

Each time NXTKEY is called, the next sequential Key in the "F" array is returned. This is based on the range specified using IONLYR, IONLYC, & IONLYZ.

As soon as the Key goes out of the bounds specified by these parameters, or the end of the "F" array is reached, a zero is returned. The first access should return the first record with the proper row, column & Z dimensions. If any of these parameters are set to zero, then the Keys to all records in the range of the remaining parameters will be output.

Valid parameter possibilities:
1. IONLYR=Ø, IONLYC=N, IONLYZ=2Ø returns all records with column #N.

2. IONLYR=M, IONLYC=N, IONLYZ=2Ø returns the record with row M, column N.

3. IONLYR=Ø, IONLYC=Ø, IONLYZ=2Ø or Ø returns all records.

4. IONLYR=Ø, IONLYC=Ø, IONLYZ=N returns all records for all rows of columns J, where G(J)=N.

con't on next page

Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
NEWKEY	N/A	N/A	NODATA	NEWKEY	N/A	N/A
			REPØØ2	NXTKEY		
			REPØØ3	NXTKEY		
			REPØØ4	NXTKEY		
			REPØØ5	NXTKEY		
			REPØ1Ø	NXTKEY		
			REPØ11	NXTKEY		
				NEWKEY		

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
N/A		IONLYR integer *4	contains the desired row number	BLOCKB	Each call only returns the key to one record. In order to get all records w/in the desired range, NXTKEY should be used until it returns a zero. The keys are returned in ascending order; i.e. row 1 always before row 2, column 1 always before column 1, etc.
		IONLYC integer *4	contains the desired column number		
		IONLYZ integer *4	contains the desired third dimension		
		I	returns the number of the column		

(con't on next page)

Software Name: NXTKEY
Type: Function
Software Author: H. Hinman, C. Martin
Person in charge of maintenançe: S, Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
NEWKEY initializes the Key variables, IROW & ICOL, to zero, and returns a zero.							

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
			being returned		
		integer *4			
		J	returns the number of the row being returned		
		integer *4			

Software Name: PARAMS
Type: Common block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 9/16/77

Purpose of Routine	Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	Called By (Entry)	Functions Accessed	Files Referenced
This common block contains: (1) a numeric value for each parameter word (IPARMS) (2) a minimum value for each parameter word (MINVAL) (3) a maximum value for each parameter word (MAXVAL) (4) a value for each parameter word to be used in default. (IDEFLT) These are all initialized in BLOCKD. IPARMS are set in RDPARM, checked against MINVAL and MAXVAL, and, if not consistent with these, the default values in IDEFLT are used. The other subroutines use this common block to obtain the values for certain of the input parameters.	N/A	N/A	N/A	BLOCKD GETDAT GETLST NEWPAG NODATA RDPARM RDTITL REPØØ2 REPØØ4 REPØØ5 REPØ11 TABS WRTROW	N/A	N/A	N/A

Error Codes/Messages Line #	Generated Message	Arguments Name	Arguments Function	Commons Referenced	Comments
N/A		IPARMS	Numeric values of the parameter words Integer array, length = 5Ø	N/A	New parameter words may be specified by adding a minimum, maximum, and default value to the appropriate arrays or changing the old ones. The parameter names may be changed in WORKA, and the default values should be inserted in the SPACE array of WORK B. In addition, any of the utilitie
		MINVAL	Minimum values of each of the parameter words Integer array, length = 5Ø		
		MAXVAL	Maximum values of each of the parameter words		

Cont on NEXT PAGE

(con't on next page)

Program Name: PCRAMS
Name: Common Block
Software Author: R. Hinman, C. Martin
Person in Charge of Maintenance: S. Masiello
Date Last Revised: 9/16/77

Cont.

Purpose of Routine

Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
		Integer array, length = 5Ø			or report generators which will use the new word/s must be updated. All of these subroutines are listed in the column 'CALLED BY'. Each of these routines contains a COMMON statement, which must be updated if the new word~~s~~ will be used in that routine.
		IDEFLT	Default values of each of the parameter words		
		Integer array, length = 5Ø			

Software Name: RDPARM
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine	Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	Called By (Entry)	Functions Accessed	Files Referenced
Starts a loop to call COMPIL and PARSE to read the input file on logical unit INPUT. When the first non-blank character in an input line is a special character it stops looping. It checks the parameters against their maximum and minimum values, and if either is exceeded, sets the respective parameter to its default value. It then reads the report title into the array 'REPORT' in common block 'TITLES' and returns.	N/A	COMPIL	COMPIL PARSE	FPDREP	N/A	N/A	INPUT (Set to #1 in FPDREP) #15

	Error Codes/Messages Line #	Generated Message	Arguments Name	Arguments Function	Commons Referenced	Comments
9Ø1	**RDPARM**	UNEXPECTED END OF FILE	INPUT Integer * 4	Tells what logical unit to use for input file.	WORKA WORKB PARAMS TITLES	When RDPARM stops looping, it returns control to FPDREP, which assumes all input specifications for the desired report have been read. Any further input is expected to be in the specific format required for the given report type.
9Ø2	**RDPARM**	PARAMETER SPECIFICATION ERROR				
	AAAA	ALLOWED MINIMUM = (I15) ALLOWED MAXIMUM = (I15) SPECIFIED VALUE = (I15) STANDARD DEFAULT NNNNN USED, EXECUTION CONTINUING				

Software Name: RDTITL
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
Reads column titles, keying on '@' to distinguish separate lines of titles. Centers these lines, and loads them into TITLE (64∅). The line read by RDTITL, from logical unit INPUT, is output, as is, to logical unit #15.	N/A	N/A	N/A	REP∅∅4 REP∅∅5 REP∅11	N/A	N/A	INPUT (defaults to #1) #15

Error Codes/Messages Generated Line #	Message	Arguments Name	Function	Commons Referenced	Comments
8∅1	**RDTITL** ____________ (Outputs line read for column title, as is)		N/A	PARAMS TITLES	Previously the following variables were logical *1: BUFR1 TITLE Making these variables logical *1 again would remove the need for the ILBYTE and ISBYTE routines

Software Name: READIT
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
Reads the KEYth record from the master file. Returns the activity short name (DNAME), MONTH, UIC, and all the data for that activity, in that time period. The data is read into ARRAY, and put into order by the accounts, using the function N. The proper record location, KEY, is determined in EXTRCT.	N/A	N/A	N/A	EXTRCT	N/A	N	#2 (Master data file)

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
N/A		RETN logical	returns true if name of activity is blank	N/A	If the activity short name (DNAME) is blank, RETN is set to TRUE (to indicate error) and ARRAY returns all zeroes for the account data. The reason for this is the assumption tha that if the correct data is loaded into the recor record the correct activity name will also be loaded.
		KEY integer *4	index to desired record		
		ISCALE integer *4	divides data by 1;1,ØØØ; or 1 million, if set= 1, 2 or3		
		DNAME	name of activity read from		

(con't on next page)

Software Name: READIT
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
			record		
		double	precision		
		MONTH	fiscal month		
		integer	*4		
		UIC	Activity UIC #		
		double	precision		
		ARRAY	returns the account-ordered data		
		double array, IS1ZE	precision length=		
		ISIZE	# of accounts		
		integer	*4		

Software Name: REPØØ2
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/11/78

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
Generates report type 2; a report with the activities as columns & the accounts as rows, with 1 time period/page. Will put as many columns across as allowed by 8Ø character width and then continue on a new page. Allows row arithmetic to be performed. This report is the reverse of report type 4. (i.e., the rows and columns are switched)	N/A	GETLST		FPDREP	N/A	NXTKEY	*INPUT
		STORTV	KFETCH			GETCHR	
		EXTRCT					
		GETDAT					
		NEWPAG					
		COMPIL	PARSE				
		TABS					
		WRTROW					

Error Codes/Messages Generated Line #	Message	Arguments Name	Function	Commons Referenced	Comments
N/A		ISIZE integer *4	length of records in the master data file. Used to determine the number of accounts, and the number of time periods	WORKB BLOCKM BLOCKK BLOCKT PARAMS TITLES	*No files are referenced by this routine; however files are referenced by some of the subroutines called, and INPUT is passed as a parameter

Software Name: REPØØ4
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenançe: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
Generates report type 4, which is the reverse of report type 2. (the columns and the rows are switched.) There is one time period/page. The accounts & account arithmetic are in the columns. The activities are in the rows.	N/A	COMPIL	COMPIL	FPDREP	N/A	NXTKEY	*INPUT
			PARSE			GETCHR	
		RDTITL					
		NEWPAG					
		STORTV	KFETCH				
		EXTRCT					
		WRTROW					
		NODATA					
		GETDAT					

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
N/A		ISIZE integer *4	length of records in master file; used to determine the number of accounts & time periods.	BLOCKK BLOCKM WORKB PARAMS	FLAG is not initialized to any value. Could cause errors. Should only be set to .TRUE. if no data exists for a requested activity. *No files are referenced; however files are referenced by some of the called subroutines & INPUT is passed as a parameter.

Software Name: REPØØ5
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/11/78

Purpose of Routine	Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	Called By (Entry)	Functions Accessed	Files Referenced
Generates report type 5, with 1 activity/page, multiple data periods (by columns) and column arithmetic, multiple accounts (by rows) and row arithmetic. Row arithmetic is performed first over column arithmetic.	N/A	RDTITL		FPDREP	N/A	NXTKEY	* INPUT
		GETLST				GETCHR	
		STORTV	KFETCH				
		EXTRCT					
		NEWPAG					
		SPREAD					
		GETDAT					
		TABS					
		WRTROW					
		COMPIL	COMPIL PARSE				

Error Codes/Messages Generated Line #	Generated Message	Arguments Name	Arguments Function	Commons Referenced	Comments
N/A		ISIZE integer *4	length of records in master file; used to determine the number of accounts and time periods.	WORKB BLOCKD BLOCKK BLOCKM BLOCKT PARAMS	*no files are referenced; however, files are referenced by some of the called subroutines and INPUT is passed as a parameter.

Software Name: REPØ1Ø
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine	Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	Called By (Entry)	Functions Accessed	Files Referenced
Generates report type 1Ø. Specialized report used to interface with the UCAR plotting routines. One activity per page, with time periods down the page (rows) and accounts across the top (columns) Similar in content to a type 3 report, although the columns & rows are reversed. However, since this report serves as a UCAR data file, the format is very specialized, and the output itself is not used as a report.	N/A	COMPIL GETDAT STORTV GETLST WRTROW EXTRCT SPREAD RDTITL	COMPIL PARSE KFETCH	FPDREP	N/A	NXTKEY GETCHR	INPUT (defaults to #1)

Error Codes/Messages Line #	Generated Message	Arguments Name	Arguments Function	Commons Referenced	Comments
N/A		ISIZE integer *4	length of records in master file; used to determine the number of accounts & time periods.	BLOCKD BLOCKK BLOCKM WORKB PARAMS TITLES	Only of use to NCD-5, for UCAR data.

Software Name: REPØ11
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/12/78

Purpose of Routine	Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
Generates report type 11. 1 account, or account computation allowed per page; multiple time periods across (columns) with activities down the side (rows) may be used. In addition, column arithmetic (computations involving time periods) is acceptable.	N/A	GETLST RDTITL NEWPAG STORTV EXTRCT SPREAD WRTROW GETDAT COMPIL	 KFETCH COMPIL PARSE	FPDREP	N/A	NXTKEY: NXTKEY NEWKEY GETCHR	*INPUT

Error Codes/Messages Line #	Generated Message	Arguments Name	Arguments Function	Commons Referenced	Comments
N/A		ISIZE integer *4	length of records in master file used to determine the number of accounts and time periods	WORKB BLOCKD BLOCKK BLOCKM BLOCKT PARAMS	*no files referenced; however files are referenced by some of the called subroutines and INPUT is passed as a parameter.

Software Name: ROUND
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/12/78

Purpose of Routine	Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	Called By (Entry)	Functions Accessed	Files Referenced
Rounds double precision numbers by adding or subtracting .5 as necessary. In DROUND, the number is then truncated to an integer. However, in DDROND, further calculations are required, and the double precision form must be retained until after these calculations. These numbers are double precision. Any truncation is performed outside of this routine.	DROUND DDROND	N/A	N/A	WRTROW	DROUND DDROND	N/A	N/A

Error Codes/Messages Line #	Generated Message	Arguments Name	Arguments Function	Commons Referenced	Comments
N/A		DARAY double precision array, length = LENGTH	inputs numbers to be rounded	N/A	There is no entry point for rounding any but double precision values.
		D2ARAY double precision array, length = LENGTH	returns DARAY, rounded, in DDROND		
		IARAY (see next page)	returns DARAY, rounded & integerized in DROUND		

- Software Name: SPREAD
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/11/78

Purpose of Routine

If the data type is requested as 'SPREAD', this routine is used for REPØØ5 and REPØ11. The data is multiplied by a factor (contained in the array FACTOR) and then divided by IDENOM. IDENOM is an input parameter, and defaults to 12, to spread yearly data into monthly. It could be set to 4, for example, to spread yearly data into quarters.

SPREAD keys on account codes to determine how to spread the data. Therefore, the development of account codes & the modification of SPREAD should be coordinated. See BLOCKM

Entry Points	Routines Called		Called By		Functions Accessed	Files Referenced
	Subroutine	(Entry)	Name	(Entry)		
N/A	N/A	N/A	REPØØ5 REPØ11	N/A	INTGER N	N/A

Error Codes/Messages	Generated	Arguments		Commons	
Line #	Message	Name	Function	Referenced	Comments
N/A		INDATA	data to be spread, double precision array, length =ISIZ	BLOCKM	The factors used are initialized to 1 in both REPØØ5 and REPØ11, and therefore, are not really used at present. Possibly it would be better to allow the factor to be entered as input to avoid re-compiling to change the factors.
		MONTH integer	specifies which element of factor to be used *4		

(con't on next page)

Software Name: SPREAD
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/11/78

(con't)

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
		ITYPE integer	data type *2		
		FACTOR real array, length	contains 'period' factors =12		
		IDENOM integer	divisor *4		
		OUTDAT array, length= ISIZ	returns spread data. double precision		
		ISIZ integer	number of accounts *4		

Software Name: SSRØØ6
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/11/78

Purpose of Routine

Used to set up heading of page. Given fiscal year and fiscal month, passes back array which holds alpha numeric data.

Example:

```
  CALL SSR006 (76,1,OUTPUT)
will return:

  OUTPUT= "31 July 1975"
```

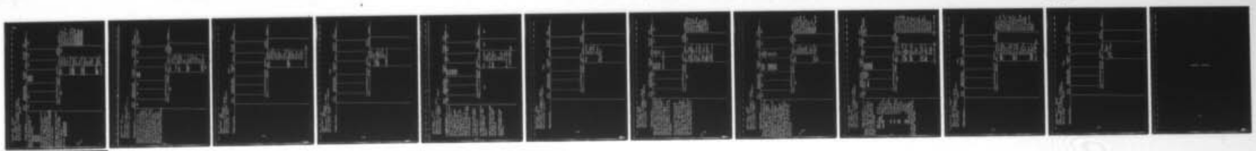
(i.e., for fiscal year 76, the first month was July, 1975)

Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
N/A	N/A	N/A	NEWPAG	N/A	N/A	#15

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
5Ø1 9Ø1	**ILLEGAL DATE CONVERSION, MONTH= NNNN**(SSRØØ6)**	IYR integer *4	Input-2 digit value of year	N/A	Previously, the following variables were logical *1: NUMBER BUFFER OUTPUT Using these as logical *1 eliminates the need for the ILBYTE and ISBYTE routines
		MONTH integer *4	Input-2 digit value of month (range= 1 to 12)		
		OUTPUT Real array, length= 5	Array which passes back the alphanumeric string of the date. (see example in purpose)		

AD-A056 055 OFFICE OF THE COMPTRO_LEF (NAVY) WASHINGTON DC F/G 9/2
MINIGAP GENERALIZED ANALYSIS PACKAGE. A TOOL FOR AIDING MANAGEM--ETC(U)
APR 78 S MASIELLO
UNCLASSIFIED MGAP-PG-01 DOD/DF-78/003A NL

2 OF 2
AD
A056055



END
DATE
FILMED
8 -78
DDC

Software Name: STORTV
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine

Used to initialize tables and pointers for use with the following routines:

- ISTORE
- DSTORE
- KFETCH
- DFETCH

The purpose of these routines is to recover an item without searching. ITEMS are stored in tables (using ISTORE,DSTORE) using random hashing and chaining; they are retrieved (KFETCH,DFETCH) in the same manner.

STORTV must be used first (once) before storing or fetching. Failure to do so could cause infinite looping.

See also: ISTORE/DSTORE
KFETCH/DFETCH

4.65

Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	Called By (Entry)	Functions Accessed	Files Referenced
N/A	N/A		FPDØØ1 GETDAT INITLZ	N/A	N/A	N/A

Error Codes/Messages Line #	Generated Message	Arguments Name	Arguments Function	Commons Referenced	Comments
N/A		TABLE 2	initialized to contain 2,3,... L, Ø	N/A	STORTV must be used for each different set of tables. For each set, TABLE 2, TABLE 3 and FREE must be <u>unique and only accessed or changed through these STORE & FETCH routines</u>.
		integer	*2 array,		
		length =L			
		TABLE 3	initialized to all zeroes		
		integer	*2 array,		
		length =L			
		L	used to dimension TABLE 2 & TABLE 3		
		integer	*4		
		FREE	initialized to 1		
		integer	*2		

Software Name: TABS
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 1/10/78

Purpose of Routine	Entry Points	Routines Called Subroutine	 (Entry)	Called By Name	 (Entry)	Functions Accessed	Files Referenced
Used just before row is sent to WRTROW to be output. Will insert IHTAB (#) words (4 characters/word) between the first 4 letters in ARRAY (the first word) and the fifth letter (the beginning of the second word), to perform horizontal tabbing. Will write IVTAB (#) blank lines to IOUTPT to perform vertical tabbing (i.e., if TABS is also called after WRTROW, vertical tabbing will be performed after the row is output. However, must also be called before WRTROW if horizontal tabbing is to be performed.)	N/A	N/A	N/A	REPØØ2 REPØØ5	N/A	N/A	IOUTPT (defaults to #15)

Error Codes/Messages Line #	Generated Message	Arguments Name	 Function	Commons Referenced	Comments
N/A		ARRAY integer	Input-contains row label *4, length =LEN1	PARAMS	
		LEN1 integer	Input-used to dimension ARRAY *4		
		IHTAB integer	Input-contains # of "words" to move row label to the right *4		
		(con't on next page)			

Software Name: TABS
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
		IVTAB integer *4	Input-contains # of lines to skip before writing "STUB" (the actual row label)		
		STUB	Output-contains ARRAY, with IHTAB(#) words between the 1st		

(con't on next page)

Software Name: TABS
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

	Entry	Routines Called		Called By		Functions	Files
Purpose of Routine	Points	Subroutine	(Entry)	Name	(Entry)	Accessed	Referenced

Error Codes/Messages	Generated	Arguments		Commons	
Line #	Message	Name	Function	Referenced	Comments
		integer *4, length = LEN2	and 2nd word.		
		LEN2 integer *4	Input - used to dimension STUB		

' Software Name: TITLES
Type: Common block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 11/15/77

Purpose of Routine

To store information used in producing the heading for each report. In BLOCKD, UNITS is initialized to contain blanks, 'AMOUNTS IN THOUSANDS' or 'AMOUNTS IN MILLIONS'. One of these options will be used depending upon the scale. Also in BLOCKD, TITLE is initialized to blanks. In RDPARM, the user-submitted title of the report is read into REPORT (up to 36 characters).

In RDTITL, each column heading is loaded into TITLE, properly spaced & centered.

In REPØØ2, RDTITL is not used and the column headings are loaded into TITLE within this routine.

NEWPAG uses the previously loaded variables of TITLES to produce the headings for each report except type 1Ø.

Since REPØ1Ø produces its own heading, it accesses TITLES directly.

Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
N/A	N/A	N/A	NEWPAG RDTITL RDPARM REPØØ2 REPØ1Ø BLOCKD	N/A	N/A	N/A

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
N/A		TITLE	to contain column headings	N/A	N/A
		Real array, length =64Ø			
		REPORT	to contain the one line, user-submitted, title of the report		
		Real array, length =9			

(con't on next page)

Software Name: TITLES
Type: Common block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 11/15/77

Cont.

Purpose of Routine	Entry	Routines Called		Called By		Functions	Files
	Points	Subroutine	(Entry)	Name	(Entry)	Accessed	Referenced

Error Codes/Messages Generated		Arguments		Commons	
Line #	Message	Name	Function	Referenced	Comments
		UNITS Real array, length=6 X 3	contains the wording for the 3 choices of scaling		

Software Name: WORKA
Type: Common Block
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine

This common block contains special characters, digits, the alphabet, and the four character abbreviation of each input parameter word. These are initialized in BLOCKD, stored into hashing tables in INITLZ, and used in COMPIL, GETCHR and RDPARM to break down the input parameter list for use by MINIGAP.

T1 is the array which contains these symbols. T3 contains a pointer to the beginning of a synonym list. The value of this pointer is obtained by hashing the symbol into T3. If synonyms are needed, T3 points to the beginning of the synonym chain in T2. In storage, TFREE points to the next available space in T1.

Entry Points	Routines Called		Called By		Functions Accessed	Files Referenced
	Subroutine	(Entry)	Name	(Entry)		
N/A	N/A	N/A	BLOCKD	N/A	N/A	N/A
			COMPIL	COMPIL		
			INITLZ	N/A		
			RDPARM	N/A		
			GETCHR (Function)			

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
N/A		T1 integer array, length = 1Ø24	List	N/A	New parameter words may be specified by adding these to T1, and making change in WORKB, PARAMS, and associated subroutines (see PARAMS, comments)
		T2 integer *2 array, length = 1Ø24	Chain		
		T3 integer *2 array, length = 1Ø24	Pointer		
		TFREE integer *2 element	pointer		

Software Name: WORKB
Type: Common block
Software Author: H. Hinman, C. Martin
Person in charge of maintenançe: S. Masiello
Date Last Revised: 9/16/77

Purpose of Routine	Entry Points	Routines Called Subroutine	Routines Called (Entry)	Called By Name	Called By (Entry)	Functions Accessed	Files Referenced
This common block contains the numeric values for the symbols of T1 (see WORKA). For example, the letter 'A' has no numeric value, so SPACE(31) = Ø, corresponding to T1(31)=1HA, and SPACE(24)=3, corresponding to T1(24)=1H3. In the elements corresponding to the input parameters, the default values are stored in BLOCKD. COMPIL changes these, as determined by its interpretation of the input list, and then the values are available for use by the four report types.	N/A	N/A	N/A	BLOCKD COMPIL RDPARM REPØØ2 REPØØ4 REPØØ5 REPØ11	N/A COMPIL PARSE N/A N/A N/A N/A N/A	N/A	N/A

Error Codes/Messages Line #	Generated Message	Arguments Name	Arguments Function	Commons Referenced	Comments
N/A		SPACE Double array, 1Ø24.	list which contains numeric values corresponding to symbols of T1. precision length=	N/A	When new parameter words are added, or the old ones changed the default values should be stored in the appropriate element of SPACE. (see WORKA and PARAMS, comments

Software Name: WRTROW
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Purpose of Routine

Writes a row of a report to IOUTPT. The row label is contained in STUB, and the data in the array DATA. The code contained in INDX determines the output, as follows:

INDX < Ø	invalid
INDX=Ø-9ØØ, 91Ø &up	the row is output as is
INDX=9Ø1	outputs a line of hyphens, the row, & a line of equals
=9Ø2	...hyphens, the row
=9Ø3	...hyphens, a line of equals
=9Ø4	...hyphens
=9Ø5	...the row, a line of equals
=9Ø6	...the row
=9Ø7	...equals
=9Ø8,9Ø9	CALL NEWPAG

To insure proper tabbing, TABS must be called prior to the call to WRTROW. (see TABS)

Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced
N/A	ROUND	DROUND	REPØØ2	N/A	N/A	IOUTPT (defaults to #15)
		DDROND	REPØ11			
	NEWPAG	N/A	REPØØ4			
			REPØØ5			
			REPØ1Ø			

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
N/A		STUB	input for label of row integer *4 array, length=LEN1	PARAMS	FLAG causes an asterisk to be output with the row. Later output from NODATA states that data for these activities is not available. However, this capability is only used by report type 4.
		LEN1	length of STUB integer *4		
		DATA	input for the numbers in the row double precision array, length= 15		Previously, STUB and IBUF were logical *1 variables. Added last parameter,
		LEN2	number of elements of DATA		
			(con't on next page)		(con't on next page

Software Name: WRTROW
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called Subroutine	(Entry)	Called By Name	(Entry)	Functions Accessed	Files Referenced

Error Codes/Messages Line #	Generated Message	Arguments Name	Function	Commons Referenced	Comments
			to output		ITITLE, to WRTROW to pass to NEWPAG. Previously, the variable concerned in NEWPAG was not redefined and was assumed to still contain the proper title. For ease in debugging and for more clarity, this variable is non redefined each time in NEWPAG.
		integer *4			
		FLAG	to footnote rows with no data (due to inaccurate data base)		
		logical			
		INDX	sets flags for printing hyphens, equals and/or data (see purpose)		
		integer *4			
		ITITLE	title printed		
		(con't on next page)			

Software Name: WRTROW
Type: Subroutine
Software Author: H. Hinman, C. Martin
Person in charge of maintenance: S. Masiello
Date Last Revised: 2/9/78

Cont.

Purpose of Routine	Entry Points	Routines Called		Called By		Functions Accessed	Files Referenced
		Subroutine	(Entry)	Name	(Entry)		

Error Codes/Messages Generated		Arguments		Commons Referenced	Comments
Line #	Message	Name	Function		
			by NEWPAG. integer *4 array, length= 1∅		

SECTION 5. OVERLAYS

OVERLAYS

The use of overlays was necessary to reduce the working size of MINIGAP. All of the utility routines, and the main routine, with the exception of INITLZ and RDPARM, are in the main segment. INITLZ and RDPARM are in one overlay and the five report generators are in the other five overlays. The six overlays and the main task constitute the MINIGAP system.

The overlays were set up using the Interdata Task Establisher (TET). The system routine IFETCH must be called before any of the routines in an overlay can be fetched by a FORTRAN call. Before a successive fetch to an overlay, that overlay must be rewound. It could be possible to remove the block data subroutine and FPD001 from the main segment and place these in overlays. However, an attempt to do this has generated error messages from TET. In the interest of time, this attempt was abandoned before knowing whether it actually could be done. (If it were constrained only by program logic, it could be done, since these are both only used once, at the start of a MINIGAP run.)

If necessary, MINIGAP could probably be separated into more overlays. This would require a certain amount of caution to insure that everything could be accessed at the proper time.

INDEX

INDEX, cont.